3-2021

An Analysis of Stability in Software Resources Data Report (SRDR) Computer Software Configuration (CSCI)

Trevor J. Violette

AN ANALYSIS OF STABILITY IN SOFTWARE RESOURCE DATA REPORT (SRDR) COMPUTER SOFTWARE CONFIGURATION ITEMS (CSCI)

THESIS

Trevor J. Violette, 1st Lieutenant, USAF

AFIT-ENV-MS-21-M-280

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENV-MS-21-M-280

AN ANALYSIS OF STABILITY IN SOFTWARE RESOURCE DATA REPORT (SRDR) COMPUTER SOFTWARE CONFIGURATION ITEMS (CSCI)

THESIS

Presented to the Faculty

Department of Systems Engineering and Management

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Cost Analysis

Trevor J. Violette, BS

1st Lieutenant, USAF

February 2021

AFIT-ENV-MS-21-M-280

AN ANALYSIS OF STABILITY IN SOFTWARE RESOURCE DATA REPORT (SRDR) COMPUTER SOFTWARE CONFIGURATION ITEMS (CSCI)

Trevor J. Violette, BS

1st Lieutenant, USAF

Committee Membership:

Dr. Jonathan D. Ritschel
Chair

Lt Col Scott T. Drylie, PhD
Member

Dr. Edward D. White
Member

AFIT-ENV-MS-21-M-280

Abstract

This research studies cost and schedule stability in programs that utilize Software Resource Data Report (SRDR) reporting standards. We find software programs at the Computer Software Configuration Item (CSCI) level show much lower levels of stability than previously published DoD stability research that focused on aircraft. A comparison of software development methods found little to no difference between Agile and Plan Driven methodologies. Critical Success Factors (CSF) were identified from prior literature and used to examine CSCIs from the SRDR dataset. Focusing on schedule or cost resulted in different variables showing significance. A CSCI is more likely to remain on budget when using a team with a low level of average experience and being judicious in your contractor selection. A CSCI is more likely to finish on schedule when a team has an average level of experience and Boeing is used as the primary contractor. A CSCI is more likely to remain on budget and on schedule when Lockheed Martin is the lead contractor and the CSCI is programmed in any language other than C. This research can be used by program managers and cost analysts to identify the critical success factors that can be utilized in the Department of Defense software environment to create trade off space between cost and schedule.

Acknowledgments

I would like to express my sincere appreciation to my faculty advisor, Dr. Daniel Ritschel, for his guidance and support throughout the course of this thesis effort. The insight and experience were both certainly appreciated.

Trevor J. Violette

Table of Contents

Page

List of Figures

Page

List of Tables

Page

AN ANALYSIS OF STABILITY IN SOFTWARE RESOURCE DATA REPORT (SRDR) COMPUTER SOFTWARE CONFIGURATION ITEMS (CSCI)

I. Introduction

Background

Knowing where performance stability is likely to occur in a project helps identify the window of time in which defense procurement officials might positively affect cost and schedule outcomes. Generally speaking, stability in a project occurs when some observed aspect (e.g. financial efficiency) of the project no longer fluctuates outside of a defined range. Without knowledge of whether stability occurs in a project, DoD project managers may waste resources on projects that are unlikely to react to outside stimuli. Previous literature (Christensen and Payne 1992, Petter et al., 2014) has identified several performance stability properties in a subset of U.S. defense programs. However, software development is sufficiently unique that these prior studies yield limited insight into whether *software projects* tend to experience problems and what the proper prognosis would be. Recent high-profile software failures such as the Expeditionary Combat Support System (ECSS) and the integrated Electronic Health Record system (iEHR) have shown that the Air Force has pursued projects for many years only to have the program cancelled with little to no capability delivered (Kanaracus, 2012; Ehlay, 2013). The subtext is that the Air Force should have known the project was ill-fated. The value of this study is to establish significant indicators for those problems which may speak to similarly ill fates, so that a program manager may have the confidence to make decisive changes.

The DoD's approach to software development is shifting towards a concept called DevSecOps. The DevSecOps construct is comprised of three parts: development (Dev), security (Sec), and operations (Ops). DevSecOps introduces security earlier into the application development lifecycle and has become the industry best practice for rapid, secure software development (Department of Defense, Chief Information Officer, 2019). The DoD's movement towards DevSecOps is occurring through a new software development venture called the DoD Enterprise DevSecOps Initiative. This initiative aims to implement agile software development methodologies, where appropriate, across all military branches with the use of open-source software (Department of Defense, Chief Information Officer, 2019). In contrast to Agile methods, traditional DoD software development practices (e.g. the waterfall method) have much longer development cycles (Kannan, 2014). These longer development cycles may mask the identification of issues that are causing cost overruns or schedule delays until late in the project's lifetime. Therefore, this research will examine whether the software development methods (e.g. agile, waterfall, etc.) employed impacts the existence of cost or schedule stability in a software program at the CSCI level.

In the DoD, Earned Value Management (EVM) is used to help project managers measure project performance. One of the key indicators used to gauge the efficiency of a program's performance is the Cost Performance Index (CPI). Previous research has defined cost performance stability as when a project's CPI does not fluctuate more than 0.1 positive or negative from the point of stability until the end of the project (Christensen and Heise, 1993). According to Christensen and Payne (1992), the importance of observing CPI stability includes helping the analyst evaluate the capability

of a contractor to recover from a cost overrun. While most previous stability research has focused on cost and using EVM data, some research has used analogous methods when EVM data is not available or when cost is not the only issue of concern for a project. Nunn-McCurdy thresholds are used by Congress as an indicator of when an MDAP experiences a cost overrun. The SRDR dataset being examined is at the CSCI level while EVM data is only given at the contract and Work Breakdown Structure (WBS) level. In the absence of EVM data, these thresholds will be used to indicate breaches in stability and will be discussed in depth further in this paper. Depending on the stability results of this research, along with the aforementioned movement towards the DoD Enterprise DevSecOps Initiative, the findings here may also suggest that current software development methods should transition to an Agile (or other) approach.

Problem Statement

The purpose of this research is to examine DoD software programs at the CSCI level for the existence of stability and, if found, determine how often this occurs. The second objective of this research is to determine whether the existence of stability occurs more or less often in agile software development methodologies than when using traditional Plan Driven software development methodologies. The third objective of this research is to identify which critical success factors in DoD software programs at the CSCI level improve stability, resulting in more accurate budgets and less cost and schedule overruns. The benefits of this research are timely with the DoD increasingly focusing on the implementation of DevSecOps in software acquisitions. For an over-

budget program, having statistical evidence that a program's cost performance is unlikely to improve can inform decisions for potential reallocation of scarce resources.

Research Questions

1. What is the extent of cost and schedule stability in DoD software intensive programs at the CSCI level?
2. What differences in cost and schedule stability properties exist between software development methodologies?
3. What DoD Software program critical success factors impact the existence of cost and schedule stability?

Methodology

The Software Resource Data Report (SRDR) Dataset from the Cost Assessment Data Enterprise (CADE) portal is used to identify which CSCIs are included in the analysis. Answering the first research question necessitates a definition of stability. As EVM data does not currently exist for SRDR programs at the CSCI level, Nunn-McCurdy thresholds will be used instead. The SRDR dataset uses initial and final reports, meaning the thresholds for Original Baseline Estimates will be used. The current thresholds are 30% and greater for a Significant Breach and 50% and greater for a Critical Breach (Schwartz, 2010).

The second part of the analysis uses hypothesis tests to identify differences between software development categories. The specific hypothesis test employed will depend on whether the assumption of normality is met. ANOVA and Tukey analysis are used if the data is normally distributed while the Kruskal-Wallis and Steel Dwass tests

are used if the assumption of normality fails. The comparison analysis will be expanded to control for factors such as software methodology (Plan Driven Vs Agile) to see what differences in stability properties exist between software development methodologies. The final part of this analysis determines the program characteristics that influence stability. The analysis employs contingency tables with CSCIs grouped into either "stable" or "unstable" based on the analysis done in part one. Through a literature review, program critical success factors are identified and incorporated as variables in the contingency tables.

Scope and Limitations

Data collections relies on the information contained in submitted 3026s from CADE. This analysis will focus on SRDR CSCIs coded as either unstable or stable CSCIs. Within the CADE database, limitations will occur due to lack of complete data and reporting inconsistencies by contractors, causing some CSCIs to be excluded from the final analysis. Furthermore, findings may be limited due to limited sample size of CSCIs using Agile processes.

Thesis Overview

The following section of this research, Chapter 2, provides relevant analysis into what defines SRDR programs as well as a literature review of critical success factors in DoD software programs. Chapter 3 outlines the methodology of the research and how modern DoD software program data at the CSCI level will be used to examine possible stability properties. Chapter 4 contains all results of the analysis and any significant

findings. The last chapter, Chapter 5, summarizes the preceding chapters, states the relevance of the findings, and presents potential ideas for future research in this area.

II. Literature Review

Chapter Overview

This literature review explores the prior work conducted on cost and schedule stability research, discusses the various software developmental methodologies in use today, and identifies critical success factors in program management. First, an introduction of the Earned Value Management (EVM) System, how Cost Performance Index (CPI) and Schedule Performance Index (SPI) are used to evaluate the efficiency of programs, and how EVM measures have been used in prior research is provided. Then, alternative evaluation methods of efficiency such as the use of Nunn-McCurdy thresholds are explored. Next, the historical and modern uses of Plan Driven and Agile software development methodologies are discussed. Finally, what critical success factors are and how they differ between software, military, and civilian programs is identified.

Earned Value Management (EVM) Overview and the Importance of Stability

This section explains the literature and background information regarding the EVM system as a whole and then focuses on how CPI and SPI are used as an efficiency index. First, an introduction of the Earned Value Management (EVM) system, CPI, and SPI are provided. Then, an overview of previous research on CPI and SPI stability is presented. The overview highlights the issues this thesis addresses, including the existence of the CPI "stability rule" and the current lack of stability research in the realm of software. While this thesis takes a different approach to stability research, understanding prior stability literature is crucial to building a strong base as to the importance of cost stability in both the DoD and the civilian sector.

EVM Background and Measurements

According to The DoD EVM Interpretation Guide (EVMIG), the authoritative source for EVM interpretive guidance, EVM is used "as a program management tool to provide joint situational awareness of program status and to assess the cost, schedule, and technical performance of programs for proactive course correction" (EVMIG, 2019) Present day EVM requirements originated in 1967 with the Cost/Schedule Control Systems Criteria (C/SCSC), a list of 35 criteria that contractors had to meet when under a contract with the US Government (Fleming and Koppelman, 1998: 19). Previous stability research has focused on EVM data. The main EVM components of interest for stability research are the CPI and the SPI. The CPI comes from dividing the Actual Cost of Work Performed (ACWP) from the Budgeted Cost for Work Performed (BCWP). The SPI comes from dividing the Budgeted Cost of Work Performed (BCWP) from the Budgeted Cost for Work Scheduled (BCWS). The BCWP, also called the "earned value," is the budgeted cost received for the total work completed. The BCWS is the "planned value" and is how much budgeted value the program should have gained so far and ACWP is the actual cost that the work incurred (DCMA, 2019: 66). Table 1 defines these five components as well as other main EVM measurements that will augment the research discussed later in this paper.

Table 1. Summary of EVM Measurements (DCMA, 2019)

EVM Measurement	Meaning	Formula
BCWP	The earned value, how much budgeted cost the program has gained thus far	Sum of the budgeted cost of all completed work packages
ACWP	The actual cost of the completed work packages thus far	Sum of actual costs of all completed work packages
BCWS	The planned value, how much budgeted value the program should have gained thus far	Sum of the budgeted cost of all work packages scheduled
SPI	Schedule efficiency of a program	BCWP / BCWS
CPI	Cost efficiency of a program	BCWP / ACWP
Schedule Variance (SV)	Difference between planned and actual schedule accomplishment	BAC – BCWP
Cost Variance (CV)	Difference between planned and actual cost accomplishment	BCWP – ACWP

Cost Efficiency Metric: CPI

As shown in the chart above, CPI is a cost efficiency measure found by dividing the BCWP by the ACWP of a program (DCMA, 2019). Being an efficiency index, CPI indicates the value of work performed for every dollar spent on a particular program. An underperforming program has a CPI value of less than 1 which would mean said program is receiving less than a dollar of value for every dollar spent on the program. An overperforming program has a CPI value of more than 1 which would mean said program is receiving more than a dollar of value for every dollar spent on said program (DCMA, 2019).

The CPI is typically used to track cost performance during a program's life cycle. CPI has valuable uses when calculated with both current and cumulative data. Current CPI focuses on using the BCWP and ACWP of the desired current period (week, month, quarter, etc.) while cumulative CPI uses the BCWP and ACWP of a program from beginning to present day. One use of cumulative CPI in major defense acquisition

programs is to calculate the Estimate at Complete (EAC_{CPI}) of a particular program. The EAC_{CPI} is the estimated final cost of a program and has shown to be "the reasonable lower bound to the final cost of a defense contract" (Christensen, 1996: 7). This shows CPI can be used to provide a good estimate on the minimum likely lowest cost for a project and can help calculate what the entire project will cost the DoD at completion.

Schedule Efficiency Metric: SPI

SPI is the other measure of efficiency and is found by dividing BCWP by BCWS as shown in Table 1 above (DCMA, 2019). SPI is best used to determine when performance of a contract is declining (Abba, 2008: 29). An SPI below 1.0 indicates the program is behind schedule, while an SPI above 1.0 indicates the program is ahead of schedule. A program with an SPI of 1.0 is exactly on schedule. SPI and SV are used less often than CPI and other measures of efficiency and effectiveness due to issues with the mathematical calculations and result terms (Fleming and Koppelman, 2000; Lipke, 2003). SV and SPI give results in terms of dollars whereas schedule deviations being stated in units of time would be more meaningful for assessing the program's performance (Lipke, 2003). Additionally, because calculations for SV and SPI use budgeted numbers, SPI will always converge back to 1 at the end of the project, meaning even when a project finishes late, SPI will not show a schedule deficiency (Fleming and Koppelman, 2000; Lipke, 2003). These weaknesses led to data being examined through other methods such as Earned Schedule (ES) and schedule stability which will be examined later in this literature review.

CPI Stability and the Stability Rule

While CPI stability has been discussed by EVM practitioners and others since the late 1970s (Christensen and Payne, 1992; Petter, 2014), documented research into CPI stability was not conducted until 1990 when Kirk Payne and David Christensen began examining 26 Contract Performance Reports (CPRs) for seven different aircraft procurement contracts from the database of the Aeronautical Systems Division (ASD) (Payne, 1990: 13). Using the definition that CPI stability occurs when the CPI of a project remains within a range of less than 0.20 from the point of stability to the end of the project, their research found stability occurred at the 20 percent completion point (Payne, 1990). These results spurred further research into this area of study and led to what we know today as the modern "stability rule." Their research also showed 53% of programs exhibiting range stability at the 0% completion mark (Christensen and Payne, 1992).

The modern stability rule originated from *A Review of Cost Performance Index Stability*, by Scott Heise with guidance from Christensen (Heise, 1991). Using the same range of 0.20, Heise and Christensen were able to find stability in the cumulative CPI of 155 contracts from the Defense Acquisition Executive Summary (DAES) database with a 95% confidence interval (Christensen and Heise, 1993). An important note on their research is that generalizability to all EVM programs cannot be assumed. In 2008, Henderson and Zwikael examined a dataset of 45 small non-DoD projects dealing with information technology and construction in the United Kingdom, Israel, and Australia. Their findings, using the interval definition of stability where the difference between the final cumulative CPI and the cumulative CPI at the 20% complete point is no more than

plus or minus 0.1, showed that the contracts did not stabilize by the 20% complete point, but in fact, "the stability is usually achieved very late in the project life cycle, often later than 80 percent complete for projects in these samples" (Henderson and Zwikael, 2008, p. 9). Abba (2008) challenged these findings, stating that Henderson and Zwikael "did not use comparable criteria to select contracts from the same source data" and that the primary data had "no evidence that these disparate projects implemented EVM consistently, as on DoD contracts," (Abba, 2008: 30). While the research of Henderson and Zwikael (2008) was critiqued due to the selected data by Abba (2008), these findings still highlight the need for further research in investigating the applicability of the stability rule in other areas.

Given the conflicting stability definitions and results from Christensen and Heise (1993) and Henderson and Zwikael (2008), Petter, Ritschel, and White (2015) examined 209 development and production contracts in the DoD from 1987 to 2012 with the goal of identifying the most useful definition of CPI stability in the EVM realm. When using the range definition, their results were consistent with past research and the modern stability rule discussed previously. However, they found merit in the argument that CPI stabilizes later when using the interval definitions. Based on the research, they concluded that "the question of stability, then, is intricately tied to the definition used" (Petter et al., 2015). With this conclusion in mind, Petter et al. recommended the absolute interval definition to provide an easier to understand and more conservative final estimate.

Expanding on the research performed by Petter et al. (2015) and Henderson and Zwikael (2008), Clayson and Thal (2018) examined monthly EVM data for 136 environmental remediation projects from a United States federal agency in fiscal years

2012 and 2013. Using the absolute interval method from Henderson and Zwikael (2008), their findings showed CPI stability occurred in the contracts at a median of 41% complete point. Using the range method discussed previously and the null hypothesis being that environmental remediation projects follow the 20 percent stability rule, the resulting p-value was 0.0059. With the null hypothesis rejected, one can conclude environmental remediation projects do not follow the 20 percent CPI stability rule (Clayson, et al., 2018). After testing the data with both the interval method and the final range method, Clayson et al. (2018) preferred the interval method as it is the "Most useful method for an analyst or manager to evaluate CPI stability for current projects as the other methods require knowing the final CPI or some unknown future CPI value to determine stability" (Clayson, et al., 2018).

To summarize the literature, the CPI stability rule has been interpreted many different ways since its inception by Christensen and Heise in 1990. This rule of thumb was questioned by Henderson and Zwikael in early 2008, but that refutation was challenged by Abba in the latter part of 2008. Since then, Petter et al. (2015) have done extensive research into the different types of interpretations, concluding the absolute interval definition is the recommended definition because it is easier to understand and more conservative, both characteristics that are important to program offices as they examine the performance of contracts. Clayton et. al (2018) used the absolute interval definition of stability discussed in Petter et al. (2015) paper to conclude environmental remediation projects become stable at 41% rather than 20% as previously thought. While the research on CPI stability using EMV data has been valuable, cost is not the only part of a project that has an impact on how successful a project is judged. Stability, such as

that found in man-hours (a proxy for cost) or in schedule, can also be analyzed through different types of data other than EVM. This research will approach stability from the consideration of both cost and schedule and will use DoD software projects as the focus point.

SPI Stability

While much research has been done into CPI stability, the literature is sparse with analysis on SPI or schedule stability. When analyzing SPI stability, Earned Schedule (ES) analysis is typically employed to create a metric called SPI(t); this metric is calculated by converting EVM data to a time-based variance instead of dollar based while utilizing the BCWP and BCWS data from traditional SPI calculations (Lipke, 2003). Henderson and Zwikael (2008) analyzed forty-five overseas information technology and construction projects using the ES technique and found that, similar to their CPI stability findings, SPI(t) stability did not occur by the 20% complete point (Henderson and Zwikael, 2008). Petter et al. (2015) examined the potential existence of SPI(t) stability in DoD acquisition programs and found SPI(t) stability behaves very similar to CPI when using the range definition but stabilizes later when using the interval definition of stability (Petter et al., 2015).

Non-EVM Stability Research

As mentioned previously, using EVM data is not the only way to measure stability. Another way to look at scheduling for software programs is to distribute software development efforts across a generalized software development life cycle. The conventional industry rule-of-thumb is 15 to 20 percent toward requirements, 15 to 20

percent toward analysis and design, 25 to 30 percent toward construction (coding and unit testing), 15 to 20 percent toward system-level testing and integration, and 5 to 10 percent toward transition (Borysowich, 2005). Yang (2008) found that factors such as development type, software size, and team size have visible impacts on effort distribution patterns, showing that a generalized effort distribution pattern has its' benefits, but identifying certain factors before beginning a project can have cost benefits (Yang, 2008).

Absent of EVM data, Congress has used Nunn-McCurdy thresholds to measure cost growth of DoD programs for nearly 30 years. The Nunn-McCurdy Act was signed into law in 1982, requiring the DoD to report to Congress whenever a Major Defense Acquisition Program (MDAP) experiences cost growth in excess of certain thresholds (Schwartz, 2010). Table 2 shows the thresholds currently in use by Congress. In the event of a critical breach, the Secretary of Defense is required to conduct a root-cause analysis on the program in question. The research in this thesis will be using the Original Baseline Estimate Breach levels as a basis for stability.

Table 2. Nunn-McCurdy Breach Thresholds

	Significant Breach	Critical Breach
Current Baseline Estimate	≥15%	≥25%
Original Baseline Estimate	≥30%	≥50%

Stability Conclusion

EVM plays a crucial role in project management and relies heavily on the use of efficiency measures like CPI and SPI. CPI stability and, to a lesser extent, SPI(t) stability is used by the EVM community with different interpretations depending on the desired use of the results. Other forms of analysis can be used depending on the data available and questions asked.

Software Background

This section explains the literature and background information regarding the history and modern uses of different software development methods. First, an introduction of which developmental methods are modernly used is discussed. Then, historical uses of Plan Driven methodologies in the DoD such as Waterfall are presented. Finally, current and future uses in the DoD of methodologies such as Agile and DevSecOps will be explored. Although Plan Driven methodologies have been around for quite some time, research on how difference methodologies impact schedule and manhour stability is slim. This research intends to fill that gap in the literature.

Importance of Software

With the United States being increasingly reliant on software to execute both missions abroad and to manage the day to day operations of the largest defense enterprise in the world, the ability to continuously develop and utilize software in new and innovative ways is central to national defense (McQuade, 2019). This requires attacking the ever-changing threats to the United States with multiple different software methodologies to capture the best response to each issue (McQuade, 2019). Examining the growth in Source Lines of Code (SLOC) from the F-16A in 1974 to the latest batch of F-35s shown in Figure 1 below, shows an increase of the mean from 135 thousand SLOC to a projected 29.5 million SLOC, further underscoring the importance of using the proper methodology for each software project (Defense Science Board, 2018). This literature review will discuss software developmental methods that are currently in use by

the DoD as well as explore newer models that the DoD is transitioning to in an attempt to maintain their competitive advantage in the realm of software.

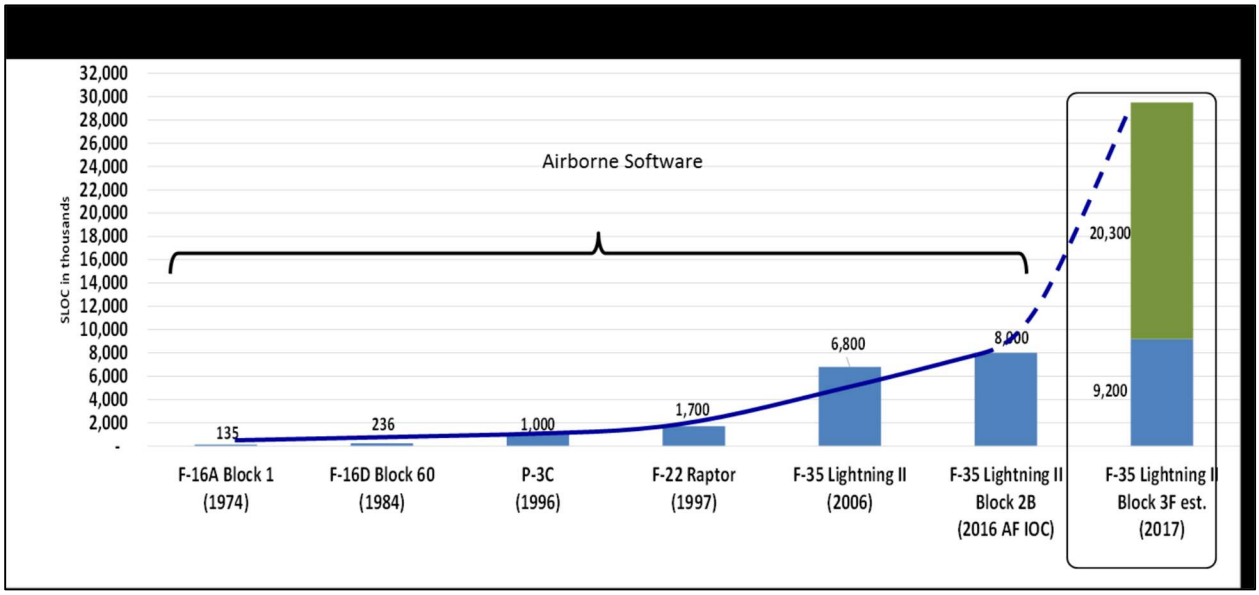


Figure 1. DoD Software Complexity and Growth: Explosive Growth of Source Lines of Code (SLOC) in Avionics Software (Defense Science Board, 2018).

Software Developmental Methods

Software Development Life Cycle (SDLC) models can be broken up into two clusters depending on how document-oriented and rigid the methodologies are. Plan Driven models, like Waterfall, Evolutionary, Spiral, Iterative, and Incremental all share the common traits of being highly documented, rigid in how they proceed from one stage to another, and typically take months to years to move from one stage to another (Akbar, Jun and Khan, 2018). Agile models, like Scrum, Extreme Programming (XP), and Dynamic System Development Method (DSDM) thrive by being flexible and rely on tacit knowledge, are less documented, and move from one state to another in short iterations called sprints which typically last no more than two to four weeks at a time (Akbar et al., 2018). For the purposes of this research, Agile methods will be grouped together.

Software industries select their development approach based on their product requirements, personnel, team skills, problem complexities, organizational needs, organization size, organizations geolocations, etc. (Chowdhury, Bhowmik, Hasan, and Rahim, 2017).

Historical Uses in DoD (Plan Driven Models)

Plan Driven Models originated in the 1970s when Winston W. Royce developed the Waterfall model (Matkovic and Tumbas, 2010). The Waterfall model relies on a high amount of documentation, clear requirements, stages that do not overlap at all, and uses sequential steps (Kannan, et al., 2014). The Waterfall method works best when quality is more important than cost or schedule or when a new version of an existing product is needed. The Spiral model keeps the high amount of documentation from the original Waterfall model, but also incorporates higher levels of risk analysis, software that is produced earlier in a program's life cycle, and is good for medium to high risk programs where requirements are likely to change more than in a program that is using the Waterfall method (Alshamrani and Bahatta, 2015). Iterative and Incremental is the closest Plan Driven Model to Agile Models and works to get a prototype of the desired product early on to allow customers more time to fine tune their requirements before the product reaches its' final build (Alshamrani and Bahatta, 2015). This reduces risk, allows more flexibility than other plan driven models, and is best for low to medium risk projects that need to get basic functionality to the end user earlier than in Spiral or in Waterfall (Alshamrani and Bahatta, 2015). Table 3 summarizes the strengths and weaknesses of common Plan Driven developmental methodologies in use today.

Table 3. Strengths and Weaknesses Comparison of Waterfall, Spiral, Incremental SDLC Models. (Alshamrani and Bahatta, 2015)

Model/Feature	**Strengths**	**Weaknesses**	**When to use**
Waterfall	• Easy to understand and implement. • Widely used and known. • Define before design, and design before coding. • Being a linear model, it is very simple to implement. • Works well on mature products and provides structure to inexperienced teams. • Minimizes planning overhead. • Phases are processed and completed one at a time.	• All requirements must be known upfront • Inflexible. • Backing up to solve mistakes is difficult, once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage. • A non-documentation deliverable only produced at the final phase. • Client may not be clear about what they want and what is needed. • Customers may have little opportunity to preview the system until it may be too late. • It is not a preferred model for complex and object-oriented projects. • High amounts of risk and uncertainty, thus, small changes or errors that arise in the completed software may cause a lot of problems.	• When quality is more important than cost or schedule. • When requirements are very well known, clear, and fixed. • New version of existing product is needed. • Porting an existing product to a new platform

Spiral	• High amount of risk analysis. • Software is produced early in the software life cycle. • Strong approval and documentation control. • Additional functionality can be added at a later date. • Project monitoring is very easy and effective. • Concerned people of a project can early review each phase and each loop as well because of rapid prototyping tools. • Early and frequent feedback from users • Suitable to develop a highly customized product. • Provides early indication of insurmountable risks.	• Cost involved in this model is usually high. • Risk assessment expertise is required. • Amount documentation required in intermediate stages makes management of a project very complex. • Time spent for evaluating risks for small or low-risk projects may be too large. • Time spent for planning, resetting objectives, doing risk analysis, and prototyping may be excessive. • Project's success is highly dependent on the risk analysis phase.	• For medium to high risk projects. • When risk evaluation and costs are important. • When significant changes are expected. • When users are not exactly sure what their needs.

Incremental/ Iterative	• Develop high-risk or major functions first. • Risk is spread across smaller increments instead of concentrating in one large development. • Lessons learned at the end of each incremental delivery can result in positive revisions for the next increment. • Customers get important functionality early and have an opportunity to respond to each build. • Each release delivers an operational product. • Initial product delivery is faster. • Reduces the risk of failure and changing the requirements.	• Requires good planning and design. • Requires early definition of a complete and fully functional system to allow for the definition of increments. • The model does not allow for iterations within each increment.	• On low to medium-risk projects. • A need to get basic functionality to the market early • On projects which have lengthy development schedules. • On a project with new technology, allowing the user to adjust to the system in smaller incremental steps rather than leaping to a major new product. • When it is high risky to develop the whole system at once.

Future Uses in DoD (Agile Models)

As previously discussed, Agile development is an umbrella term for all current lean and flexible methodologies in use today. Agile methodologies expanded on techniques used in iterative and incremental processes to increase flexibility and the speed in which products are delivered to consumers (Kannan, Jhajharia, and Verma, 2014). Agile methods focus on personalizing the methods used to best suit the project requirements. Agile works best when customers are heavily involved in the process, a working prototype can be developed early in the software's life cycle, individuals are

able to be highly motivated and self-organized, and the Agile team is able to adapt to changing requirements quickly and competently (Kannan, et al., 2014).

Software Conclusion

This section introduced the relevant software developmental methods to this research. This research will attempt to fill the literature gap on the impact of these different software methodologies on cost and schedule stability. This is an important gap to fill as software is becoming increasingly important in defense acquisition program success or failure.

Critical Success Factors Background

A critical success factor (CSF) is a high-level goal that is imperative for a business to meet. CSF's are factors that must be vital to the organization's success, beneficial to the company or department, be synonymous with a high-level goal, and link directly to the business strategy (Vanderbyl and Kobelak, 2007). Factors will vary across businesses and industries; however, the identification of them is essential to maintaining focus for the duration of a project. CSFs are used in this research to formulate the independent variables for analysis of cost and schedule stability in software CSCIs. This segment of the literature review is broken down into three sections: CSFs as they relate to military projects in general, CSFs in software specific applications, and CSFs as they relate to the civilian sector.

Military Projects

Critical success factors in defense development projects historically revolved around five key criteria:

Table 4. Critical Success Factors in Defense Development Projects (Tishler, Dvir, Shenhar and Lipovetsky, 1996)

1. The more critical the perceived need, the greater chance of success
2. Amount of customer follow-up
3. Clear and reasonable scope
4. Clear requirements early in the project
5. Professional qualifications and high motivation of the development team

These CSFs were found by analyzing 110 defense projects executed in the 1980s and 1990s. 20 measures of success were derived for each project with the above five being the most common. (Tishler, Dvir, Shenhar and Lipovetsky, 1996) As will be seen below, with slight differences, these CSFs have significant overlap with software projects today.

Software Specific

According to Nasir and Sahibuddin (2011), there are 26 critical success factors that influence the successful completion of software projects. Of these, five were observed in more than 50% of the researched literature. They are:

Table 5. Critical Success Factors in Software (Nasir and Sahibuddin, 2011)

Critical Success Factors in Software	**Observance Frequency**
1. Clear requirements and specifications	60.5%
2. Clear objectives and goals	55.8%
3. Realistic Schedule	53.5%
4. An effective project manager	53.5%
5. Support from top management	51.2%

User/client involvement was 6th in number of observations, meaning all five of the CSFs identified from defense development projects in the 1980s-1990s are observed in the top CSFs for software projects from the 1990s-2010s.

Commercial Perspective

In private industry, many sectors attempt to understand the vital elements required to maintain a successful business. Looking specifically at the biotech industry, a survey conducted by Vanderbyl, Kobelak, and Biotechnol (2007) analyzed 247 biotechnology companies across Canada documenting universal critical success factors (Vanderbyl and Kobelak, 2007). The top five categories of critical success factors shed light on influential values in the commercial sector:

Table 6. Top Five Categories of Critical Success Factors (Vanderbyl and Kobelak, 2007)

1. Knowledge assets including the IP and internal and external company databases.
2. Use of resources from internal R&D products to entering of foreign markets.
3. External environment ranging from government support, industry clusters to resources networking.
4. Funding focusing on marketing conditions, management expertise, and development of products.
5. Recruitment investigating the human resource issues in the nation, from lack of qualified candidates to available resources to compete for these candidates.

In a more specific examination of software specific projects, Sudhakar (2011) attempts to synchronize the values essential to an effective software project. The main objective of Sudhakar's (2011) article is to attempt to identify the critical success factors which are essential to any software development projects. He identifies the following CSFs:

Table 7. Critical Success Factors Essential to Any Software Development Project (Sudhakar, 2011)

1. Communication in project	2. Quality control
3. Top management support	4. Client acceptance
5. Clear project goals	6. Accuracy of output
7. Reliability of output	8. Reduce ambiguity
9. Project planning	10. Maximize stability
11. Teamwork	12. Realistic expectations
13. Project team coordination	14. User involvement.

Project management, product, team, and communication factors are also identified as important categories of success factors for software projects (Sudhakar, 2011). The author recognizes that there must be flexibility in determining how these factors can be applied to specific projects; however, he highlights that every attempt must be made to keep each of these concepts in mind for the duration of the work.

In 2018, Garousi, Tarhan, Pfahl, Coşkunçay, and Demirörs applied an empirical approach to identifying which CSF's are the most pertinent to a project's success (Garousi, Tarhan, Pfahl, Coşkunçay and Demirörs, 2018). The empirical analysis was based on the data via an online questionnaire-based survey in the Turkish software industry, in which the dataset included data from 101 software projects. They found that the top three CSF's are: (1) project team's experience with the software development methodologies, (2) project team's expertise with the task, and (3) project monitoring and controlling (Garousi, et al., 2018). They recognize that this work can be useful for software managers at all levels to help prioritize the improvement opportunities within their respective organizations.

Also in 2018, Lavazza, Morasca, and Tosi examined what CSFs impact productivity in commercial software programs. Specifically, they examined the impact of the primary programming language on new and enhancement software projects. This study found that programming language had a statistically significant impact on productivity for new programs but did not for enhancement programs (Lavazza, et al., 2018). While stability is not the same as productivity, examining the primary programming languages for the CSCIs in this dataset could provide program managers another tool to improve cost and schedule stability in their programs.

Critical Success Factors Conclusion

While critical success factors vary across organizations, they remain important to identify and manage successfully to ensure timely completion of a project. Successful software programs, whether defense oriented or civilian, share commonalities in the CSF identified. The CSFs identified through this literature review will be utilized in this research to examine cost and schedule stability properties in defense software programs at the CSCI level.

Chapter Summary

EVM is a critical tool that can be used to explore when a program stabilizes in cost and schedule, but it is not the only method to look at stability research. A sizeable body of literature shows that stability occurs and that it can be measured in numerous ways. Examining the various software developmental methodologies (plan driven vs. agile) and their impact on cost and schedule stability will fill a current gap in the literature. Additionally, through the identification of CSFs in this literature review, testable independent variables can be incorporated into the stability models utilized in this research. The next chapter will discuss in more detail the methodological approach to this research.

III. Methodology

Chapter Overview

Chapter 3 provides detailed information about the data and the methodology utilized to analyze the data. First, it explains the source of the data, characteristics of the data, and discussion on what data from the initial dataset is excluded. Next, this chapter explains the definition of stability that will be used to determine whether a CSCI is defined as "stable" or "unstable" for our analysis. Then, we outline the hypothesis testing we utilize in our comparison analyses between software developmental methodologies. Finally, program critical success factors are identified and incorporated as variables in the contingency table analysis to test their impact on the existence of stability in software DoD programs at the CSCI level.

Data

Data for this analysis comes from the Defense Automated Cost Information Management System (DACIMS). DACIMS is nested inside the Cost Assessment Data Enterprise (CADE) database which is the central repository for Software Resource Data Reports (SRDRs) as well as other datasets. SRDRs capture software effort, size, and schedule metrics for usage as historical reference points in future cost estimates and enterprise resource planning efforts (OSD CAPE, 2019). SRDRs are reported by contractors for all major software development contracts in Acquisition Category I and IA programs, Major Defense Acquisition Programs (MDAP), and Major Automated Information System (MAIS) programs following Milestone A (OSD CAPE, 2019).

The initial dataset consists of 4499 SRDRs spanning from Jan 2001 to Oct 2019, representing a broad range of programs at the CSCI level across numerous platform types. In order to provide repeatable analysis in the future, the SRDR Verification and Validation (V&V) Guide steered the selection criteria from initial to final dataset. To create a dataset suitable for growth analysis, the dataset must exhibit three main criteria. First, each data point must be analyzed and tagged as "good" in accordance with established V&V quality tags. To receive a "good" tag, the data point must not have missing data, must not be an interim report with unverified hours, must have hours allocated at the proper level, and must not have any unverified anomalies. This removed 3032 datapoints, leaving behind 1462. Next, each datapoint must be the first initial submission or final submission for a project, further reducing the testable dataset by 362 down to 1100. Finally, each Initial SRDR must have a corresponding Final SRDR, culminating in the final growth analysis dataset of 335 datapoints. A further 30 datapoints were removed before schedule stability analysis due to an analysis of peak staff and hours resulting in an impossible schedule tag being applied by the SRDR V&V Guide (Lanham, et al., 2018). Table 8 depicts the exclusion criteria and accompanying number of datapoints utilized for this research.

Table 8. Dataset Exclusions

Category	**Number Removed**	**Number Remaining**
Initial Data Set	0	4499
Any data point not reaching the acceptable level to be good based on V&V quality tags	3037	1462
Any data point that is not an initial or final report	362	1100
Any data point that does not have a data pair *-- 335 Data Points is the starting point for Man Hours Analysis*	665	335
Any data point with the impossible schedule tag	27	308
3 more added to IS tag due to further analysis *-- 305 Data Points is the starting point for Schedule Months Analysis*	3	305
Further Inclusion/Exclusions based on specific RQ needs		

Stability Determination

The initial stability analysis is performed in two parts. The Man-Hours variable will be used as a proxy for cost analysis while the Schedule-Months variable will be used for schedule analysis. Prior stability research has leaned on the definitions of stability used by Christensen (Christensen, 1993). These definitions were created using EVM data. Because of the lack of EVM data available for DoD software programs, SRDR data was used instead, resulting in a new definition of stability needing to be created. This definition must be usable when an analyst only has the initial and final reports for a program available. Of Christensen's definitions for stability, only the range stability rule, which states a program is stable if the CPI at the point of stability is within 0.1 +/- of the final CPI, could potentially be used as a starting analogous definition for this research.

Stability for both dependent variables, Schedule-Months and Man-Hours, is calculated by using the formula $\frac{Final\ Value}{Initial\ Value} - 1$. To determine stability without EVM data, the Nunn-McCurdy Act thresholds are used. The specific Nunn-McCurdy thresholds employed are a 30% change +/- for a Significant Breach and a 50% change +/-

for a Critical Breach. CSCIs that fall within these thresholds are deemed stable and datapoints that fall outside the thresholds are deemed unstable. 335 datapoints are used for the cost stability analysis and 305 datapoints are used for the schedule stability analysis. As explained in Table 1, the difference of 30 datapoints in the respective analyses is due to the Impossible Schedule tag.

Data Characteristics for Software Development Methods Comparison

The impact of software development methods on cost and schedule stability is the next part of the analysis. To accomplish this requires further delineation of the data. Two types of analyses are completed: 1) an "Upper Group" comparison of Agile to Plan Driven methods and 2) a "Subgroup" analysis of Agile, Evolutionary, Incremental, Iterative, Spiral, and Waterfall methods. Plan Driven methodologies encompassed any methodology that developed software in a sequential manner rather than an iterative manner such as Agile. The Subgroup analysis contains any methodology identified in the SRDR with an N of five or more that only used one software methodology.

Table 9 provides an overview of the characteristics of the datapoints in the final SRDR Dataset used for this portion of the research. The cost stability dataset consists of 335 datapoints, 18 used Agile and 308 used Plan Driven methodologies. When analyzing whether software methodology impacted the likelihood of a CSCI showing cost stability throughout its lifecycle, nine datapoints were removed from the Upper Group analysis due to the CSCI using a combination of Agile and Plan Driven methodologies in their project. A total of 30 datapoints were removed from the Subgroup cost stability analysis due to the CSCIs in question using a combination of different software methodologies.

For the Upper Group and Subgroup analysis using the Schedule Months variable, seven and 27 datapoints respectively were removed.

Table 9. Dataset Characteristics

	Man Hours	Schedule Months
Starting Datapoints	335	305
Upper Group Breakdown		
Agile	18	16
Plan Driven	308	282
Unfit for Analysis	9	7
Subgroup Breakdown		
Agile	18	16
Evolutionary	5	5
Incremental	81	71
Iterative	42	35
Spiral	59	56
Waterfall	100	95
Unfit for Analysis	30	27

Comparative Analysis of Software Developmental Methods

The premise of research question two is to determine whether the various software developmental methods have statistically significant differences in cost or schedule stability. To answer these questions, several statistical tests, such as the Shapiro-Wilk, Kruskal-Wallis, and Steel-Dwass tests are used to analyze the data through hypothesis testing. The Shapiro-Wilk test is used to determine normality for both the Man-Hours and Schedule-Months dependent variables, leading to the rejection of the null hypothesis that normality existed within these two populations. With the assumption of normality being rejected, non-parametric testing is used to test for relations between the Man-Hours variable and each examined category. These non-parametric tests are repeated for the Schedule-Months variable and each examined category. The Kruskal-

Wallis test is used to determine whether compared category locations were statistically significantly different from one another. Finally, for the categories that showed statistically significantly different locations, the Steel-Dwass test was used to identify which compared medians were statistically different from one another.

Contingency Table Analysis

For the third and final research question, the impact of independent variables from the dataset on the dependent variables, Man-Hours, and Schedule-Months, is tested. This will be done through the use of contingency tables for each dependent variable. A two-way contingency table is used to show the relationship between two categorical variables based on the data observed. When determining statistical significance for the contingency table analysis, the chi-square distribution is used. Each test will use a 2x2 table and utilizes the same hypothesis test. The null hypothesis will be that the two classifications are independent while the alternative hypothesis will be that the two classifications are dependent. A failure to reject the null means the two variables are not statistically related to each other. A rejection of the null shows a statistical dependency between them and will be further examined. The two-way contingency analysis examines the variables listed below in Table 10.

Table 10. Independent Variables Examined

Potential Independent Variables	**Source/Data Location**
Requirements Volatility Initial	CSF Lit/SRDR Dataset
Requirements Volatility Filtered	CSF Lit/SRDR Dataset
New/Upgrade	SRDR Dataset
Contractor	SRDR Dataset
Team Experience Level	CSF Lit/SRDR Dataset
Service	SRDR Dataset
Programming Language	SRDR Dataset

The initial amount of datapoints used for contingency analysis remained 335 for Man-Hours and 305 for Schedule-Months. Remaining datapoints with the Impossible Schedule tag were removed from the dataset in between analysis for Man-Hours and Schedule-Months for each independent variable analysis.

For Requirements Volatility, analysis was done two ways. In both, 42 datapoints were removed from analysis due to having no requirement volatility data. Then, for the first requirements volatility analysis, the remaining datapoints were recoded as such: nominal was coded as 0, very low as 1, low as 2, medium as 3, high as 4, and very high as 5. This resulted in an analysis of 293 datapoints for Man-Hours and 270 for Schedule-Months. The second requirements volatility analysis removed any datapoint without a cardinal number resulting in 231 datapoints for Man-Hours and 213 for Schedule-Months.

The New/Upgrade variable describes whether the CSCI is in a New program or one going through an Upgrade. The New/Upgrade variable required removal of seven datapoints due to those datapoints being coded as both New and Upgrade, leaving 328 datapoints for Man-Hours and 303 for Schedule-Months. For the Contractor variable, no datapoints required removal.

For the Team Experience Level variable, 55 were removed for no data and improper coding, leaving 280 datapoints for Man-Hours and 258 for Schedule-Months. These datapoints were put onto a 1-5 scale and bucketed as Low, Average, and High for 1-2.99, 3-4, and 4.01-5 scale, respectively.

For the Service variable, one datapoint was removed due to the Marines only having one datapoint. This left 334 datapoints for Man-Hours and 304 for Schedule

Months. Lastly, the Programming Language had 14 datapoints removed due to datapoints with multiple primary coding language being used or improper coding. This left 321 for the Man-Hours variable and 291 for the Schedule-Months variable. Table 11 provides an overview of the datapoints in the final SRDR Dataset used for this portion of the research.

Table 11. Independent Variable Exclusions

Potential Independent Variables	**Number Removed**	**Man-Hours Remaining Datapoints**	**Schedule-Months Remaining Datapoints**
Requirements Volatility Initial	42	293	270
Requirements Volatility Filtered	104	231	213
New/Upgrade	7	328	303
Contractor	0	335	305
Team Experience Level	55	280	258
Service	1	334	304
Programming Language	14	321	291

Chapter Summary

This chapter reviewed the methodological approach to analyzing the SRDR dataset. The discussion of the data and data characteristics offers insights into how and why the dataset used for this research provided an effective basis for research into cost and schedule stability. Next, the definition of what constitutes “stable” and “unstable” CSCIs in our analysis was explained. Then, categories and sub-categories used for comparison analysis were highlighted to capture the intent of the research. The following chapter will provide a detailed look at the results and analysis of the hypothesis testing used for research question two and the comparative analysis process used for research question three.

IV. Analysis and Results

Chapter Overview

This chapter presents the results from applying the Chapter III methods and is broken down into three sections, mirroring the research questions posed in Chapter I. The first section provides an overview of the dataset and then categorizes each software CSCI as stable or unstable. The second section examines the impact software methodology has on the presence of schedule and cost stability in the dataset. The final section analyzes which independent variables have statistically significant impacts on the existence of stability in the Man-Hours and Schedule-Months dependent variables.

Data Characteristics

All data utilized in the statistical analysis conducted in this research was gathered from the Cost Assessment Data Enterprise (CADE) system from Software Resource Data Reports (SRDRs). Table 12 depicts the exclusion criteria and accompanying number of datapoints utilized for the research.

Table 12. Dataset Exclusions

Category	**Number Removed**	**Number Remaining**
Initial Data Set	0	4499
Any data point not reaching the acceptable level to be good based on V&V quality tags	3037	1462
Any data point that is not an initial or final report	362	1100
Any data point that does not have a data pair 335 Data Points is the starting point for Man Hours Analysis	665	335
Any data point with the impossible schedule tag	27	308
3 more added to IS tag due to further analysis 305 Data Points is the starting point for Schedule Months Analysis	3	305
Further Inclusion/Exclusions based on specific RQ needs		

Existence of Stability

The initial analysis focused on what constitutes stability in CSCIs that use SRDR as well as what percentage of CSCIs can be considered "stable" and "unstable" for the extent of this research. The number of stable CSCIs differed depending on what definition of stability was used. Without the existence of EVM data for this dataset, the Nunn-McCurdy thresholds of 30% and 50% respectively for Significant and Critical Breaches are used. Descriptive statistics show what number of CSCIs show stability with the two thresholds above. In total, 130 of the 335 CSCIs examined for Man-Hours stability and 156 of the 305 CSCIs for Schedule-Months did not meet the threshold for a Significant Breach. 175 of the 335 CSCIs examined for Man-Hours stability and 219 of the 305 CSCIs for Schedule-Months did not meet the threshold for a Critical Breach. Table 13 shows the number and percentage of CSCIs that show stability for each Dependent Variable.

Table 13. Existence of Stability

Man-Hours	**Number of Stable CSCIs**	**Percentage of Stable CSCIs**
Stable without a Nunn-McCurdy Significant Breach	130	38.80
Stable without a Nunn-McCurdy Critical Breach	175	52.24
Schedule-Months		
Stable without a Nunn-McCurdy Significant Breach	156	51.15
Stable without a Nunn-McCurdy Critical Breach	219	66.56

When looking at the distribution of stability for the Man-Hours dependent variable, the data is mostly normally distributed outside of the large amount of CSCIs showing more than triple their initial man hours used estimations. In Figure 2, the green bars show

CSCIs that did not have a Significant Breach in stability while the yellow bars show ones that did not have a Critical Breach.

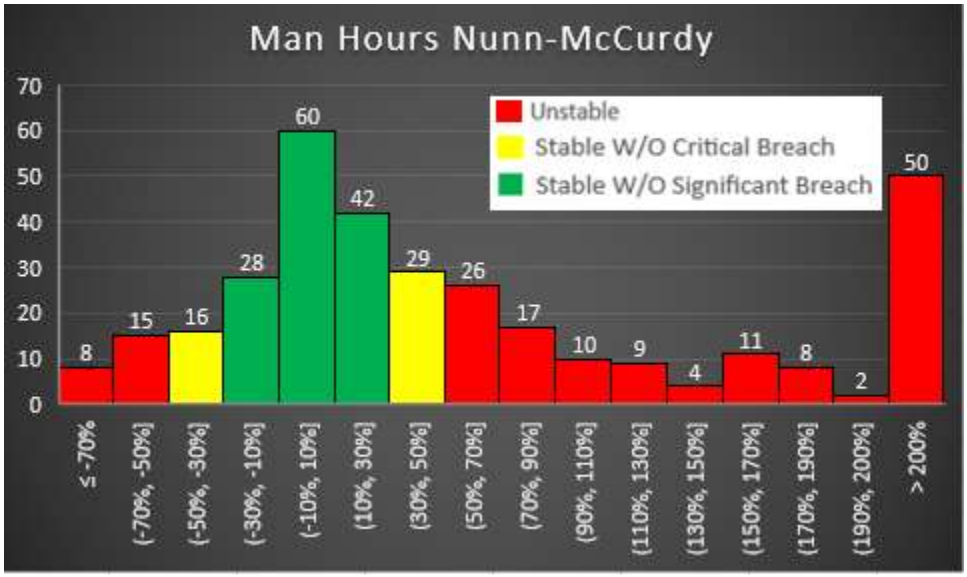


Figure 2. Man-Hours Nunn-McCurdy Thresholds

Figure 3 shows the same analysis for the Schedule-Months dependent variable. A key difference between the two variables for the presence of stability lies in the CSCIs that finished ahead of schedule or cost. For Man Hours, 23 CSCIs were “unstable” but actually finished under budget for their man hours estimations. For Schedule-Months, this drops to only four CSCIs or 1.3%. Again, in Figure 3 below, the green bars represent stable CSCIs without Significant Breach while the yellow ones show ones that did not have a Critical Breach.

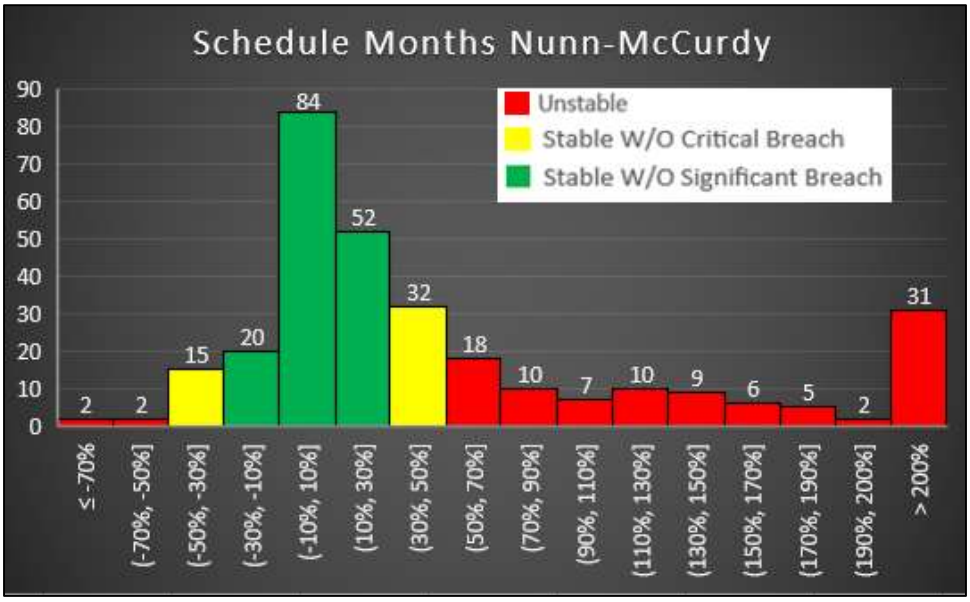


Figure 3. Schedule-Months Nunn-McCurdy Thresholds

Results of the Comparative Analysis of Software Developmental Methods

Research question one provided a definition of stability for the examined CSCIs. The next step in the research was to determine if the software developmental method used for the CSCI had any impact on the existence of this stability. This was done by first testing for normality in the two dependent variables with the Shapiro Wilk test. The Shapiro Wilk test and corresponding outputs are shown in Figure 4. With neither dependent variable showing normality at an alpha level of 0.05, non-parametric tests were used to test for relations between the two dependent variables and each examined category.

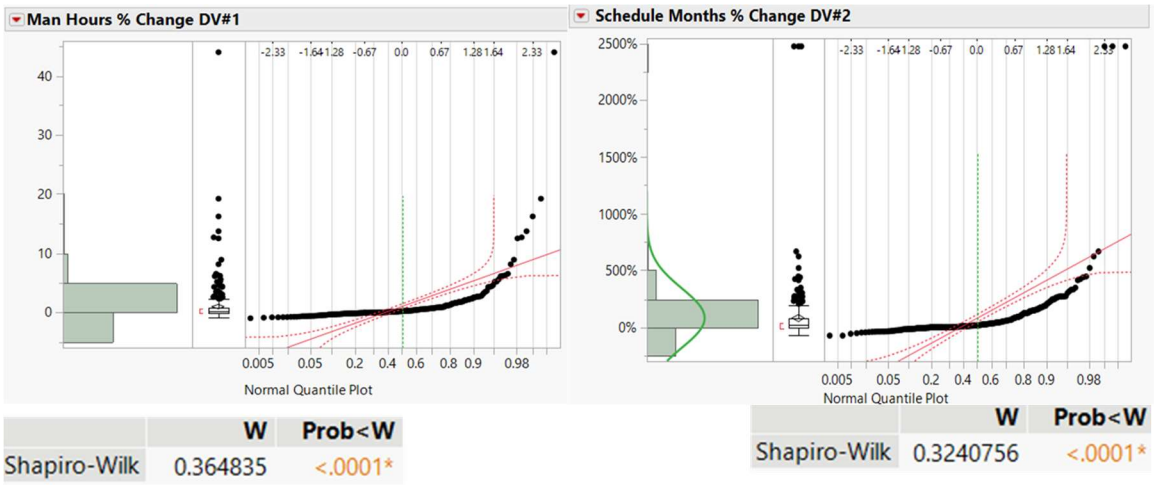
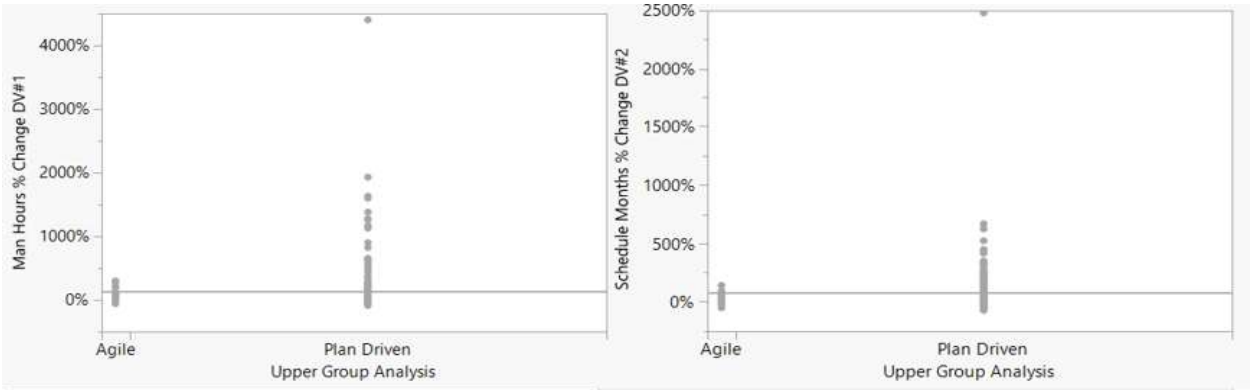


Figure 4. Shapiro Wilk Output

The first two Rank Sum tests ran between each dependent variable and the Upper-Level group did not show a statistically significant difference between software methodologies at an alpha level of 0.05 as shown in Figure 5. Because non-parametric tests are being used to examine the dataset, the potential outlier at the top of Figure 5 would have no impact on the results. With Agile being the most recent software methodology of those examined, it was expected the results would show a difference in either cost or schedule. This was not the case, meaning that the benefits to using Agile methodology may lie in the intangibles such as customer satisfaction rather than cost or schedule.



Wilcoxon / Kruskal-Wallis Tests (Rank Sums)

Level	Count	Score Sum	Expected Score	Score Mean	(Mean-Mean0)/Std0
Agile	18	3163.50	2943.00	175.750	0.566
Plan Driven	308	50137.5	50358.0	162.784	-0.566

2-Sample Test, Normal Approximation

S	Z	Prob>|Z|
3163.5	0.56602	0.5714

1-Way Test, ChiSquare Approximation

ChiSquare	DF	Prob>ChiSq
0.3218	1	0.5705

Wilcoxon / Kruskal-Wallis Tests (Rank Sums)

Level	Count	Score Sum	Expected Score	Score Mean	(Mean-Mean0)/Std0
Agile	16	2541.00	2392.00	158.813	0.443
Plan Driven	282	42010.0	42159.0	148.972	-0.443

2-Sample Test, Normal Approximation

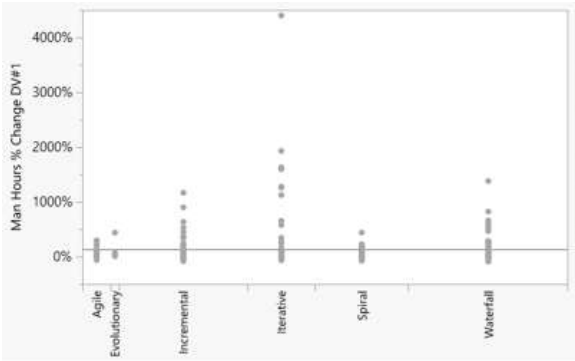
S	Z	Prob>|Z|
2541	0.44326	0.6576

1-Way Test, ChiSquare Approximation

ChiSquare	DF	Prob>ChiSq
0.1978	1	0.6565

Figure 5. Upper Group Output

The Rank Sum tests at the Subgroup level found varying results for each dependent variable. The Subgroup analysis showed no statistically significant difference between any of the listed methodologies for Man-Hours as seen below in Figure 6.



Wilcoxon / Kruskal-Wallis Tests (Rank Sums)

Level	Count	Score Sum	Expected Score	Score Mean	(Mean-Mean0)/Std0
Agile	18	2904.50	2754.00	161.361	0.413
Evolutionary	5	931.000	765.000	186.200	0.846
Incremental	81	12486.5	12393.0	154.154	0.137
Iterative	42	6554.00	6426.00	156.048	0.240
Spiral	59	8598.00	9027.00	145.729	-0.704
Waterfall	100	15191.0	15300.0	151.910	-0.150

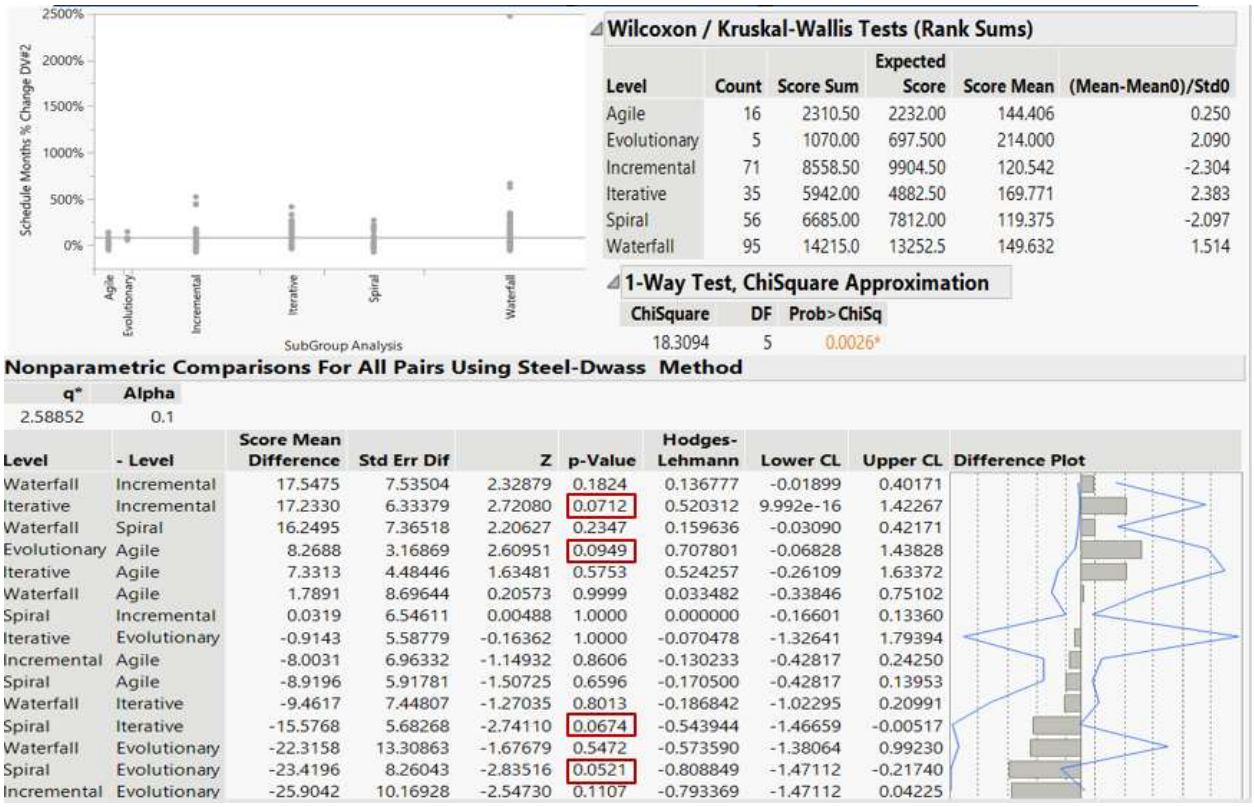
1-Way Test, ChiSquare Approximation

ChiSquare	DF	Prob>ChiSq
1.3508	5	0.9296

Figure 6. Man-Hours Subgroup Output

For the Schedule-Months variable, the global Kruskal-Wallis Rank Sum test found a statistically significant difference between software methodologies. The Steel-Dwass test must then be used as it preserves the familywise error rate (Type 1) and provides accurate p-values. The Steel-Dwass test did not find a statistically significant difference between

methodologies at an alpha level of 0.05, but it did find differences at an alpha level of 0.1. These p-values are circled in red in Figure 7 below. The four highlighted p-values show that while Agile is comparable to most Plan Driven methodologies in terms of Schedule Stability, statistically significant differences do exist within the Plan Driven subgroup.



Wilcoxon / Kruskal-Wallis Tests (Rank Sums)

Level	Count	Score Sum	Expected Score	Score Mean	(Mean-Mean0)/Std0
Agile	16	2310.50	2232.00	144.406	0.250
Evolutionary	5	1070.00	697.500	214.000	2.090
Incremental	71	8558.50	9904.50	120.542	-2.304
Iterative	35	5942.00	4882.50	169.771	2.383
Spiral	56	6685.00	7812.00	119.375	-2.097
Waterfall	95	14215.0	13252.5	149.632	1.514

1-Way Test, ChiSquare Approximation

ChiSquare	DF	Prob>ChiSq
18.3094	5	0.0026*

Nonparametric Comparisons For All Pairs Using Steel-Dwass Method

q*	Alpha
2.58852	0.1

Level	- Level	Score Mean Difference	Std Err Dif	Z	p-Value	Hodges-Lehmann	Lower CL	Upper CL	Difference Plot
Waterfall	Incremental	17.5475	7.53504	2.32879	0.1824	0.136777	-0.01899	0.40171	
Iterative	Incremental	17.2330	6.33379	2.72080	0.0712	0.520312	9.992e-16	1.42267	
Waterfall	Spiral	16.2495	7.36518	2.20627	0.2347	0.159636	-0.03090	0.42171	
Evolutionary	Agile	8.2688	3.16869	2.60951	0.0949	0.707801	-0.06828	1.43828	
Iterative	Agile	7.3313	4.48446	1.63481	0.5753	0.524257	-0.26109	1.63372	
Waterfall	Agile	1.7891	8.69644	0.20573	0.9999	0.033482	-0.33846	0.75102	
Spiral	Incremental	0.0319	6.54611	0.00488	1.0000	0.000000	-0.16601	0.13360	
Iterative	Evolutionary	-0.9143	5.58779	-0.16362	1.0000	-0.070478	-1.32641	1.79394	
Incremental	Agile	-8.0031	6.96332	-1.14932	0.8606	-0.130233	-0.42817	0.24250	
Spiral	Agile	-8.9196	5.91781	-1.50725	0.6596	-0.170500	-0.42817	0.13953	
Waterfall	Iterative	-9.4617	7.44807	-1.27035	0.8013	-0.186842	-1.02295	0.20991	
Spiral	Iterative	-15.5768	5.68268	-2.74110	0.0674	-0.543944	-1.46659	-0.00517	
Waterfall	Evolutionary	-22.3158	13.30863	-1.67679	0.5472	-0.573590	-1.38064	0.99230	
Spiral	Evolutionary	-23.4196	8.26043	-2.83516	0.0521	-0.808849	-1.47112	-0.21740	
Incremental	Evolutionary	-25.9042	10.16928	-2.54730	0.1107	-0.793369	-1.47112	0.04225	

Figure 7. Schedule-Months Subgroup Output

Contingency Table Analysis

A 2x2 contingency table analysis is used to examine relationships between two variables. Significant Relationships are identified when Pearson's chi-squared test statistic has a p-value of less than 0.10. The null hypothesis of Pearson's chi-squared test is that the two classifications are independent. If there is a failure to reject the null, the two variables are not statistically related to one another. If the null is rejected, then the

variables are dependent, and a statistical relationship exists between them. The existence of spurious relationships is possible when analyzing these results. A spurious relationship occurs when the two variables are associated, but not causally related. This can be caused by unknown mediating variables. Therefore, only highly significant results with a p-value of less than 0.01 are further analyzed while the other significant variables are observed only as potential findings.

The dataset analyzed consisted of two variables of interest, Man-Hours and Schedule-Months, as well as 31 categorical dummy variables. For the contingency table analysis performed on the Man-Hours variable, three variables were significant at an alpha of 0.10, four at an alpha of 0.05, and one at an alpha of 0.01. The full set of test results can be seen in Table 14 below.

Table 14. Man-Hours Significant Contingency Tables

Variable	**Man-Hours Significant Breach**	**Man-Hours Critical Breach**
New/Upgrade		
Req Volatility Initial 0		
Req Volatility Initial 1		
Req Volatility Initial 2		
Req Volatility Initial 3		
Req Volatility Initial 4		
Req Volatility Initial 5		
Req Volatility Filtered 0		
Req Volatility Filtered 1		
Req Volatility Filtered 2		
Req Volatility Filtered 3		
Req Volatility Filtered 4		
Req Volatility Filtered 5		
Contractor Bae Systems		
Contractor Boeing		
Contractor General Dynamics	*	***
Contractor Lockheed Martin		*

Contractor Northrop Grumman		
Contractor Raytheon		
Team Experience Level Low	**	**
Team Experience Level Average	**	
Team Experience Level High		
Service Air Force		
Service Army	*	
Service Navy		
Programming Language Ada		
Programming Language C	**	
Programming Language C#		
Programming Language C++		
Programming Language C/C++		
Programming Language Java		
Total Significant Contingency Tables	**5**	**3**
Table Legend: * p-value < 0.10 ** p-value < 0.05 *** p-value < 0.01		

Table 14 test results suggest that whether a CSCI is New or an Upgrade has no significant impact on whether the program at the CSCI level will stay on budget. These results also suggest that, regardless of which way requirement volatility is calculated (Requirement Volatility Initial or Requirement Volatility Filtered), it has no significant relationship with the cost stability of the SRDR CSCIs in question. When observing the relationship between the contractor overseeing the program and the cost stability of said program at

the CSCI level, three contingency tests showed significance, with one of them flagging at an alpha level of 0.01. The test for Lockheed Martin suggests a CSCI is *more* likely to finish within 30% of the initial Man-Hours budget when Lockheed Martin is the lead contractor. The contingency test for General Dynamics suggests a CSCI is *less* likely to finish without a Significant Breach in Man-Hours if General Dynamics is the lead contractor. When expanding that test to include Critical Breaches, the Chi Square test statistic becomes highly significant at an alpha level of 0.01 as shown in Figure 8 below. The odds ratio indicates that given the program is managed by General Dynamics, the odds of the program at the CSCI level finishing with a Critical Breach in Man-Hours is 2.8 times higher than if the program is managed by any other contractor.

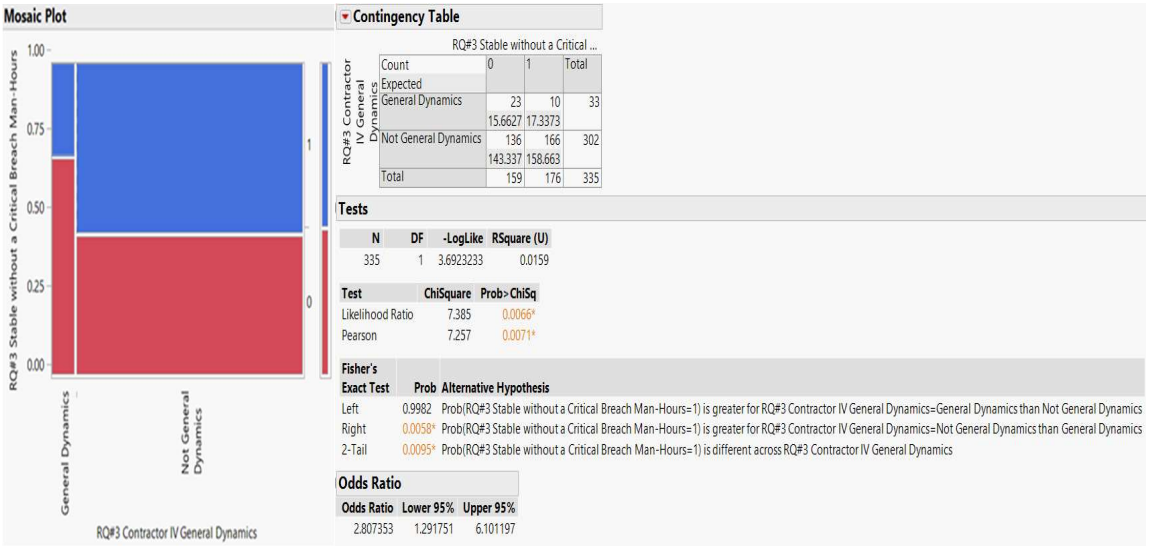


Figure 8. Critical Breach Man-Hours by Contractor General Dynamics

Examining the team experience level tests, three variables were significant at a 0.05 alpha level. The results suggest that teams with a low average experience level are *more* likely to finish without a Significant or a Critical Breach in Man Hours. This may be because a less experienced team is likely expected to take longer to complete similar

tasks as a more experienced team This would mean the less expected teams would have more flex room in their initial man hours estimates and finish within their expected man hours estimate more often. The results for when a team has an average experience level support this conclusion as well, as a team with average experience is *less* likely to finish within 30% of their expected Man-Hour budget. When looking at the results for the Service variable, they indicate a CSCI is *less* likely to finish without a Significant Breach in Man-Hours if Army is the branch in charge. Lastly, a CSCI that is programming in C instead of other programming languages is *less* likely to finish without a Significant Breach in Man-Hours.

Next, contingency table analysis is performed with the Schedule-Months variable. Six variables were significant at an alpha of 0.10, nine at an alpha of 0.05, and 12 at an alpha of 0.01. Due to the number of significant tests, the full set of test results are broken up by significant and critical breaches shown in Table 15 and Table 16, respectively.

Table 15. Schedule-Months Significant Breach Significant Contingency Tables

Variable	**Schedule-Months Significant Breach**
New/Upgrade	
Req Volatility Initial 0	*
Req Volatility Initial 1	**
Req Volatility Initial 2	
Req Volatility Initial 3	
Req Volatility Initial 4	***
Req Volatility Initial 5	**
Req Volatility Filtered 0	
Req Volatility Filtered 1	
Req Volatility Filtered 2	
Req Volatility Filtered 3	*
Req Volatility Filtered 4	***
Req Volatility Filtered 5	**
Contractor Bae Systems	

Contractor Boeing	***
Contractor General Dynamics	
Contractor Lockheed Martin	
Contractor Northrop Grumman	
Contractor Raytheon	
Team Experience Level Low	***
Team Experience Level Average	***
Team Experience Level High	
Service Air Force	
Service Army	***
Service Navy	***
Programming Language Ada	**
Programming Language C	
Programming Language C#	
Programming Language C++	
Programming Language C/C++	
Programming Language Java	
Total Significant Contingency Levels	**13**
Table Legend: * p-value < 0.10 ** p-value < 0.05 *** p-value < 0.01	

Similar to the Man-Hours analysis, the New/Upgrade variable shows no significant relationship with the Schedule-Months Significant Breach variable, indicating the chances of staying on time is not impacted by what kind of CSCI you are estimating. The requirement volatility analysis showed similar results between the Requirement Volatility

Initial and Requirement Volatility Filtered variables depending on what data was being included in the analysis. The Requirement Volatility Initial variable included as much data as possible, with volatility reported in words such as very low, low, medium, high, and very high, being recoded as 1, 2, 3, 4, and 5, respectively. The Requirement Volatility Filtered variable focused strictly on data already coded as 0,1,2,3,4, and 5. This resulted in 0 and 1 showing significance for Requirement Volatility Initial, 3 showing significance for Requirement Volatility Filtered, and 4 and 5 for both Requirement Volatility variables when looking at Significant Breaches in schedule. Table 9 suggests that a CSCI is *more* likely to finish on schedule if *more* requirement volatility is present. At first, this seems counter intuitive because a CSCI that is not having changing requirements should be more likely to understand the workload and finish on time. However, there are two scenarios where that might not be the case. Requirement volatility does not necessarily mean requirements are growing: if requirements are shrinking, a CSCI would have less work needed to be done, resulting in staying on schedule more often. Second, if a CSCI is having shifting requirements, it might be under intense scrutiny, resulting in higher pressure to finish on time. Conversely, CSCIs without changing requirements could be low priority time wise and are allowed to finish late. This would stand to reason why requirement volatility 4 for both variables, shown in Figure 9 and 10 below, show significance at an alpha level of 0.01 because those CSCIs would have above average volatility, but not necessarily to the point where a complete restart would be required.

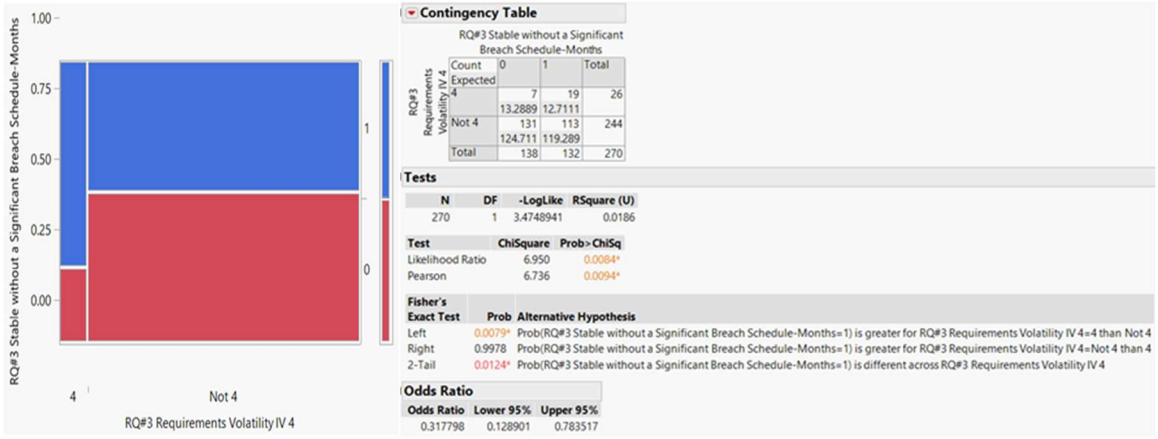


Figure 9. Significant Breach Schedule-Months by Requirements Volatility Initial 4

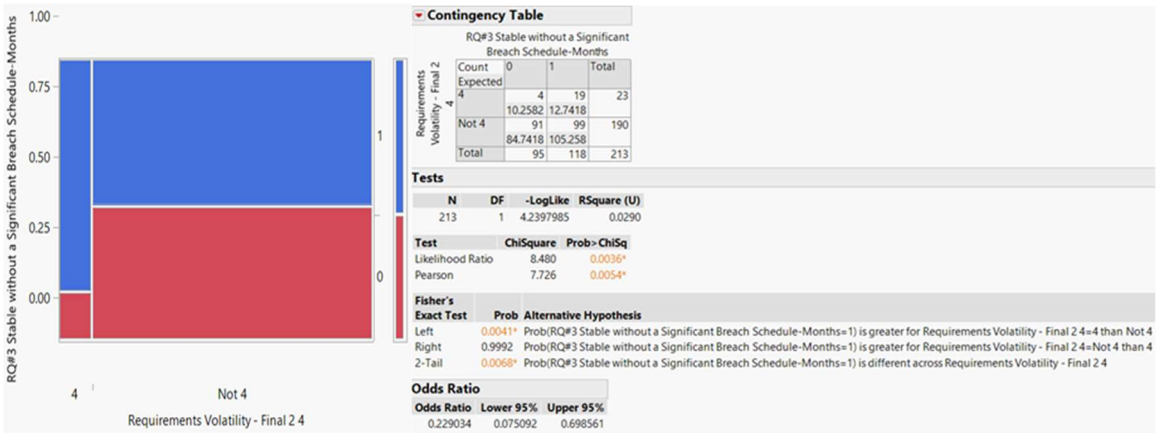


Figure 10. Significant Breach Schedule-Months by Requirements Volatility Filtered 4

When looking at Significant Breaches in schedule for the Contractor variable, only Boeing showed significance. Figure 11 shows Boeing is significantly *more* likely to finish within 30% of their initial time estimate than any other contractor DoD works with.

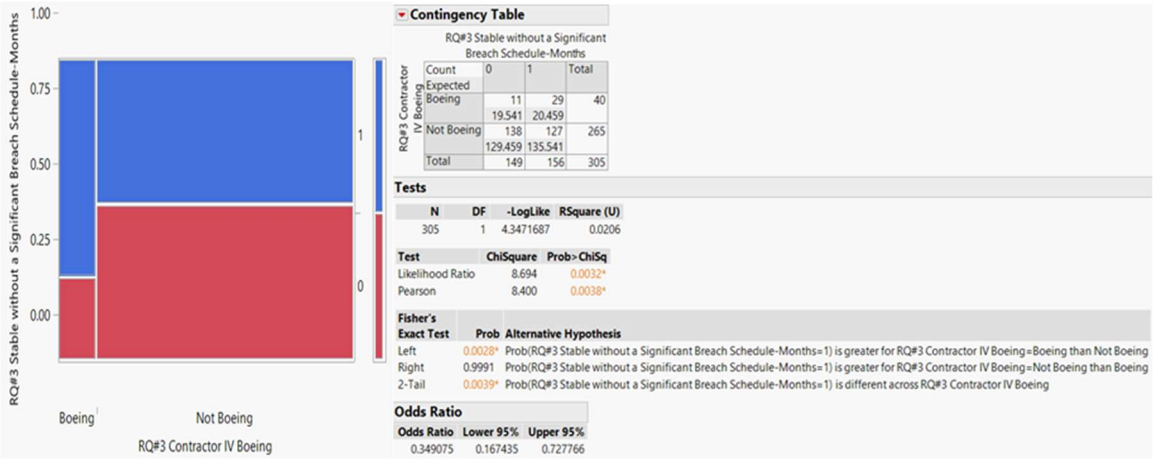


Figure 11. Significant Breach Schedule-Months by Contractor Boeing

Next, the Team Experience Level variable found significance at an alpha level of 0.01 for both low and average experience levels. These two tests show opposite Fisher's Exact Tests', as seen in Figures 12 and 13. These figures show that a project is *less* likely to finish on time if a team has a low average experience level and *more* likely to finish on time if their experience level is average. Interestingly, a highly experienced team does not show a significant relationship with finishing on time, meaning there is no benefit in terms of schedule stability of having a more experienced team past a certain extent.

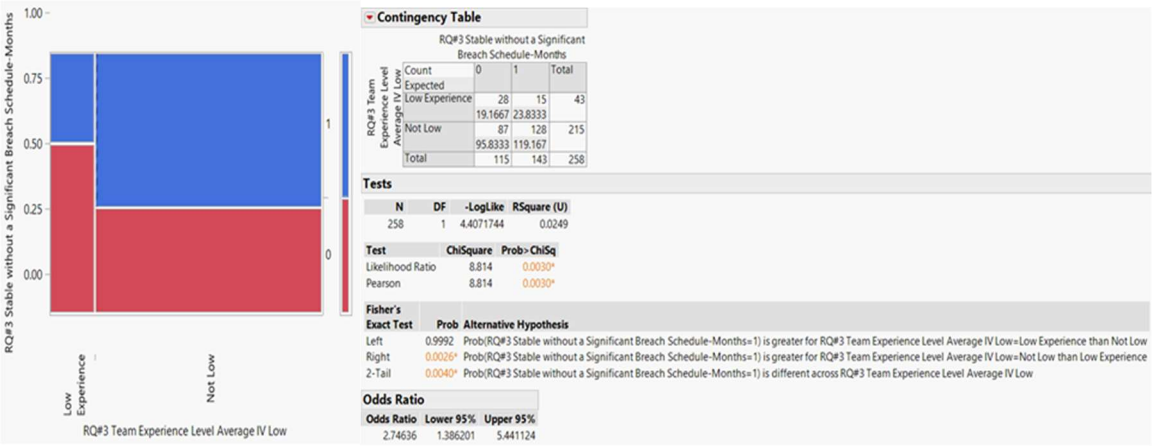


Figure 12. Significant Breach Schedule-Months by Team Experience Level Low

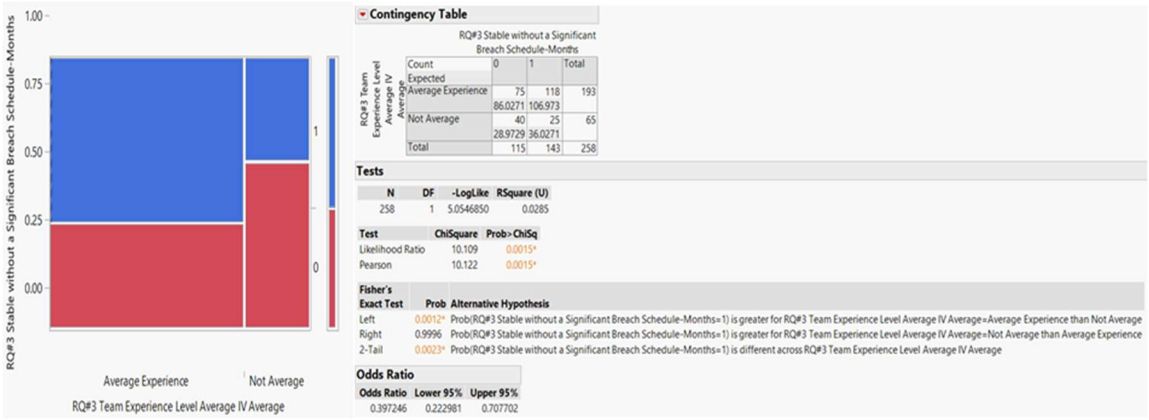


Figure 13. Significant Breach Schedule-Months by Team Experience Level Average

The last two variables of interest were Programming Language and Service. For Programming Language, only Ada showed significance at an alpha level of 0.05, but C slightly missed making the cut at 0.1049. These results suggested that a CSCI is *less* likely to finish without a Significant Breach in Schedule-Months if using C, but *more* likely if using the language Ada. This may be because Ada was originally created by the DoD, resulting in potentially more experienced Ada users than C users on DoD contracts. For Service, Army and Navy both showed significance at a 0.01 level with Fisher's Exacts Tests' opposite of what was seen with the Team Experience Level variable. Figures 14 and 15 make clear that a CSCI was *more* likely to finish on schedule with Army instead of Navy in charge.

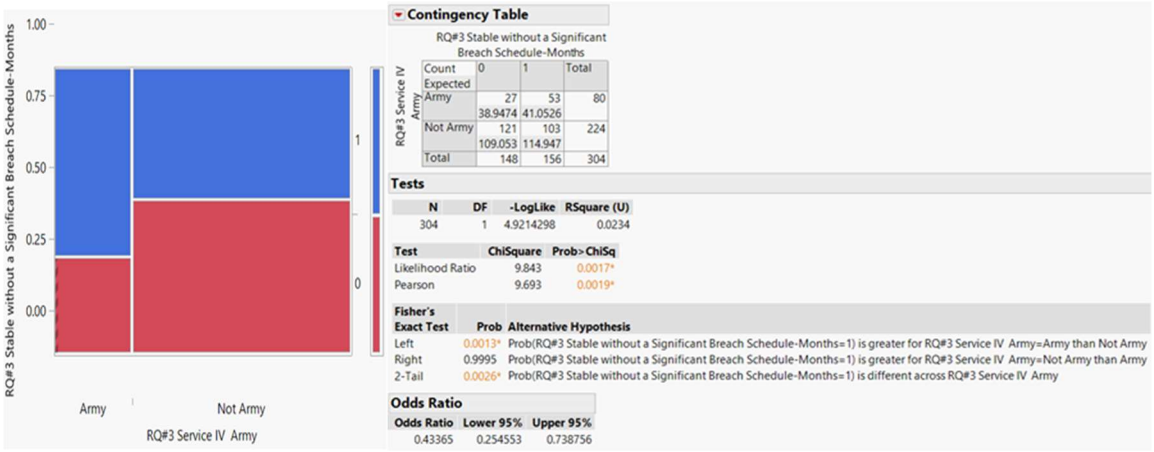


Figure 14. Significant Breach Schedule-Months by Service Army

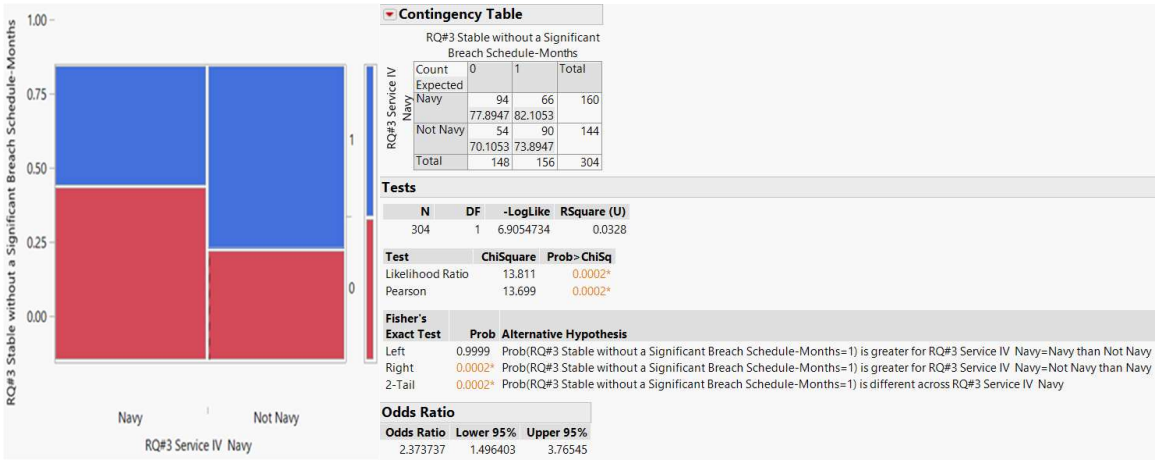


Figure 15. Significant Breach Schedule-Months by Service Navy

Table 16. Schedule-Months Critical Breach Significant Contingency Tables

Variable	Schedule-Months Critical Breach
New/Upgrade	
Req Volatility Initial 0	
Req Volatility Initial 1	**
Req Volatility Initial 2	*
Req Volatility Initial 3	
Req Volatility Initial 4	*
Req Volatility Initial 5	
Req Volatility Filtered 0	
Req Volatility Filtered 1	**
Req Volatility Filtered 2	
Req Volatility Filtered 3	
Req Volatility Filtered 4	*

Req Volatility Filtered 5	
Contractor Bae Systems	
Contractor Boeing	***
Contractor General Dynamics	
Contractor Lockheed Martin	***
Contractor Northrop Grumman	**
Contractor Raytheon	
Team Experience Level Low	*
Team Experience Level Average	***
Team Experience Level High	**
Service Air Force	
Service Army	***
Service Navy	***
Programming Language Ada	**
Programming Language C	
Programming Language C#	
Programming Language C++	
Programming Language C/C++	
Programming Language Java	
Total Significant Contingency Levels	**14**
Table Legend: * p-value < 0.10 ** p-value < 0.05 *** p-value < 0.01	

Similar to the findings in Table 15, Table 16 also suggests that a CSCI is *more* likely to finish on schedule if *more* requirement volatility is present. The tests lose significance when talking about Critical Breaches instead of Significant Breaches

regarding the Requirement Volatility variable. The loss of significance indicates that observed Critical Breaches are slightly more in line with expectations, meaning volatile requirements would have less of an impact on schedule if one was okay with finishing within 50% of initial schedule estimates rather than within 30%. At an alpha level of 0.05, the contingency test including Northrop Grumman found a project is *less* likely to finish without a Critical Breach if Northrop Grumman is the company in charge of said project. Boeing again showed significance at an alpha level of 0.01 as shown in Figure 16. Lockheed Martin (see Figure 17) joined Boeing in significance at an alpha level of 0.01. These two figures show an extremely high probability of finishing within 50% of initial estimates if a project has Boeing or Lockheed Martin as the lead contractor. Lockheed Martin did not show any relationship to the Schedule-Months Significant Breach variable while Boeing did. Lockheed Martin did show a relationship at the 0.10 alpha level for Critical Breaches in cost. This means if staying on schedule was your only concern and all else was held constant, Boeing would be a better choice as a contractor but if staying on budget was a factor as well, Lockheed Martin may be the preferred option.

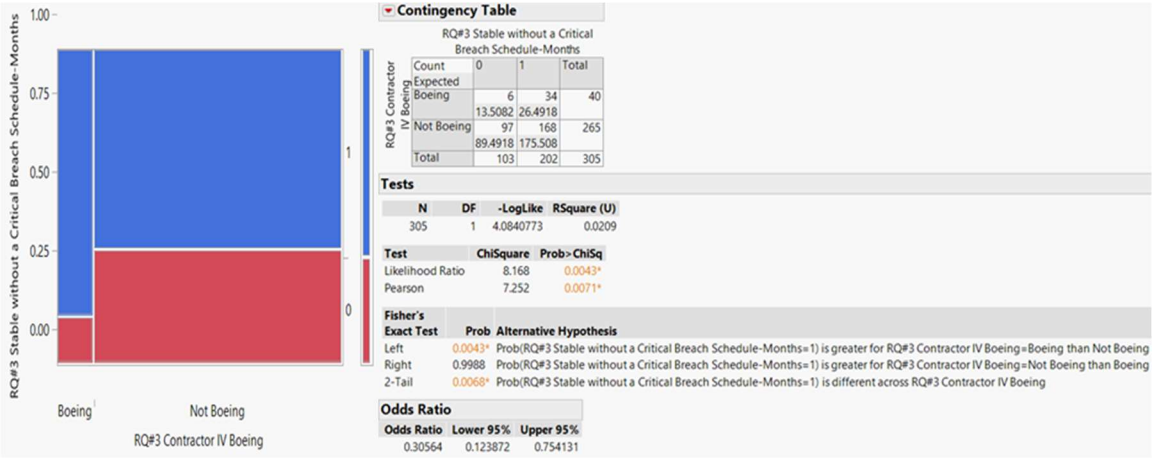


Figure 16. Critical Breach Schedule-Months by Contractor Boeing

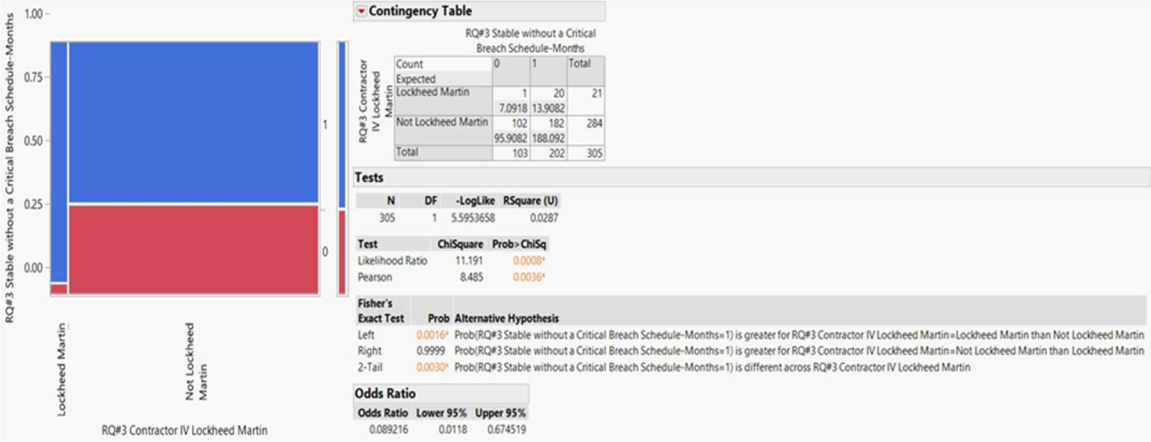


Figure 17. Critical Breach Schedule-Months by Contractor Lockheed Martin

All three team-experience variables found some level of significance at various alpha levels. Like the Schedule-Months Significant Breaches, a low experienced team is *less* likely to finish on schedule while a team with an average experience level is *more* likely to finish on schedule. Teams with a high experience level also showed to be *less* likely to finish on schedule at an alpha level of 0.05. Like Table 15, the programming language Ada showed significance at an alpha level of 0.05, showing a CSCI is better off using Ada if they want to finish without a Critical Breach in Schedule-Months. Lastly, both Army and Navy showed significance at an alpha level of 0.01 with opposite Fisher's Exact Tests', also like what was shown in Table 14. Figures 18 and 19 show that a project is *less* likely to finish on time if being led by the Navy and *more* likely to finish on time if led by Army.

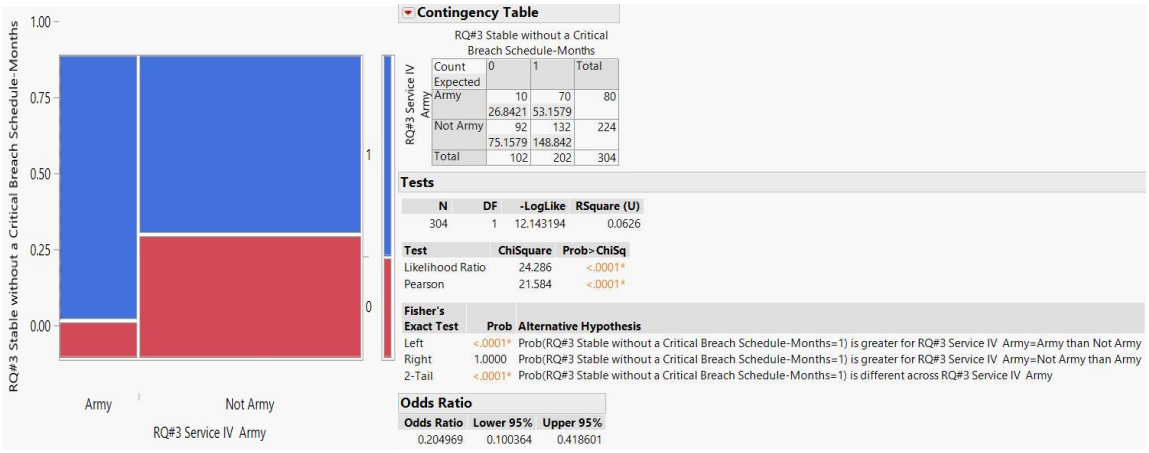


Figure 18. Critical Breach Schedule-Months by Service Army

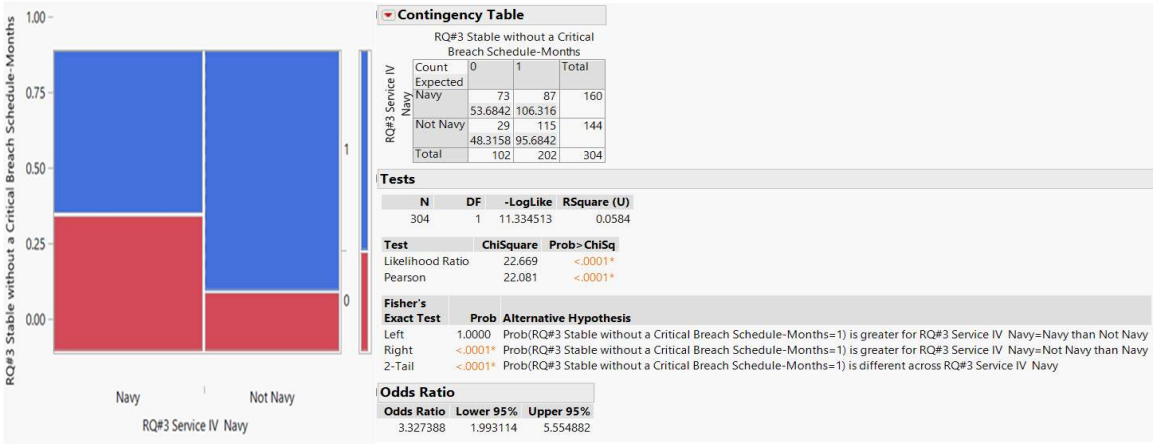


Figure 19. Critical Breach Schedule-Months by Service Navy

Contingency Table Analysis Summary

Several potential findings were examined through the results of a contingency table analysis. Relationships with the Man-Hours variable suggests that SRDR CSCIs are *less* likely to finish on budget with General Dynamics as the lead contractor, with an average experience level team, programmed in C, or if managed by the Army. SRDR are *more* likely to finish on budget if Lockheed Martin is the lead contractor or the team has a low level of average experience.

Relationships with the Schedule-Months variable suggest that SRDR CSCIs are *less* likely to finish on schedule if the team has a low level of average experience, if requirement volatility is low, if Northrop Grumman is the lead contractor, if the project is managed by Navy, or is programmed in C. SRDR CSCIs are *more* likely to finish on schedule if requirement volatility is above average, a team has average level of average experience, Lockheed Martin or Boeing is the lead contractor, managed by Army, or programmed in Ada. All other tested variables showed in insignificant relationship with the Man-Hours and Schedule-Months variables.

Multivariate Analysis

One of the weaknesses of running a contingency table analysis is the potential of omitted variable bias. If there is a missing variable, such as size, that is the true driving factor for cost and schedule stability, not including it could lead to misleading results. Looking for omitted variable bias in a dataset can be done by using a correlation matrix with all variables and the potential omitted variable. In Table 17, Estimated Source Lines of Code (ESLOC) – Final is tested against all variables used in this analysis. ESLOC – Final is used because it is the actual lines of code used in each CSCI. If stability is more likely to occur in larger programs, Table 17 would show a strong correlation between any of the dependent variables and the ESLOC – Final variable. No dependent variable has more than a 0.0470 correlation with ESLOC, indicating an unsubstantial correlation between ESLOC and schedule or cost stability. An unsubstantial correlation is also shown between ESLOC and all independent variables as the highest correlation is 0.2089.

Table 17. ESLOC Correlation Matrix

Dependent Variables	**Correlation with ESLOC - Final**
Stable without a Nunn-McCurdy Significant Breach Man Hours	0.0275
Stable without a Nunn-McCurdy Critical Breach Man Hours	0.0416
Stable without a Nunn-McCurdy Significant Breach Schedule Months	0.0275
Stable without a Nunn-McCurdy Critical Breach Schedule Months	0.0470
Independent Variables	
New/Upgrade - New	-0.0065
New/Upgrade - Upgrade	0.0065
Req Volatility Initial 0	-0.0180
Req Volatility Initial 1	-0.1164
Req Volatility Initial 2	0.0836
Req Volatility Initial 3	-0.0069
Req Volatility Initial 4	0.0479
Req Volatility Initial 5	0.0322
Req Volatility Filtered 0	-0.0180
Req Volatility Filtered 1	-0.1164
Req Volatility Filtered 2	0.0836
Req Volatility Filtered 3	-0.0069
Req Volatility Filtered 4	0.0479
Req Volatility Filtered 5	0.0322
Contractor Bae Systems	0.0144
Contractor Boeing	0.1533
Contractor General Dynamics	-0.0333
Contractor Lockheed Martin	-0.0615
Contractor Northrop Grumman	-0.0619
Contractor Raytheon	0.0808
Team Experience Level Low	0.1757
Team Experience Level Average	-0.1092

Team Experience Level High	-0.0678
Service Air Force	-0.0109
Service Army	-0.0625
Service Navy	0.0646
Programming Language Ada	-0.1283
Programming Language C	-0.0426
Programming Language C#	0.0431
Programming Language C++	-0.0489
Programming Language C/C++	-0.0001
Programming Language Java	0.2089

Chapter Summary

This chapter presented the results from applying the Chapter III methods and was broken down into three sections, mirroring the research questions posed in Chapter I. The first section provided an overview of the dataset and then categorized each software CSCI as stable or unstable. The second section examined the impact software methodology has on the presence of schedule and cost stability in the dataset. The final section analyzed which independent variables have statistically significant impacts on the existence of stability in the Man Hours and Schedule Months dependent variables. The next chapter will further discuss these results and provide the conclusions drawn from this research and analysis.

V. Conclusions

Chapter Overview

This chapter utilizes the analysis and results from the previous chapter to answer the initial research questions. These questions are answered through a dialogue that highlights their impact and potential use in current cost analysis applications. Specific results and findings are presented for each phase of the analysis. Finally, the limitations and potential future research are discussed.

Findings

This research originated with the purpose of filling the literature gap on stability in DoD software intensive programs. To do this, the definition of stability was modified to fit non-EVM data. This led to using Nunn-McCurdy thresholds and the two dependent variables, Man-Hours and Schedule-Months, as proxies for cost stability and schedule stability respectively. A Significant Breach occurs when a CSCI exceeds 30% of its initial estimate while a Critical Breach occurs when a CSCI exceeds 50% of its initial estimate. While finishing behind schedule or over budget is considered worse than ahead of schedule or under budget, any inaccuracy in budget or schedule leads to improper allocation of funds. Because of this, Table 17 includes all CSCIs that did not meet the threshold to be considered stable on both the low and high end.

Table 18. Existence of Stability

Man-Hours	Number of Stable CSCIs	Percentage of Stable CSCIs
Stable without a Nunn-McCurdy Significant Breach	130	38.80
Stable without a Nunn-McCurdy Critical Breach	175	52.24
Schedule-Months		
Stable without a Nunn-McCurdy Significant Breach	156	51.15
Stable without a Nunn-McCurdy Critical Breach	219	66.56

From the table, one can tell a project is more likely to finish on time than on budget which makes sense as schedule slips are more easily seen than budget slips. Another interesting note is that only approximately 15% of CSCIs finish between 30-50% off schedule or budget, meaning that a majority of CSCIs are either without a Significant Breach in budget or schedule or they completely blow by their initial estimates. This can be seen in Figure 20 and Figure 21 below as the second largest bin for Man-Hours and the 4th largest bin for Schedule-Months is >200%. In Figure 20 below, the green bars show CSCIs that did not have a Significant Breach in stability while the yellow bars show ones that did not have a Critical Breach.

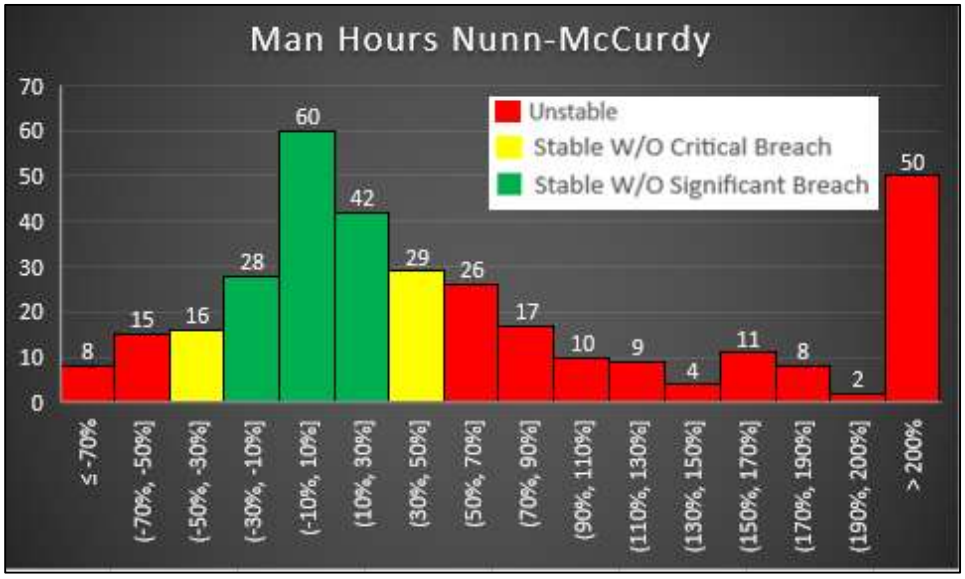


Figure 20. Man-Hours Nunn-McCurdy Thresholds

Figure 21 shows the same analysis for the Schedule-Months dependent variable. A key difference between the two variables for the presence of stability lies in the CSCIs that finished ahead of schedule or cost. For Man Hours, 23 CSCIs were "unstable" but actually finished under budget for their man hours estimations. For Schedule-Months, this drops to only four CSCIs or 1.3%. Again, in Figure 21 below, the green bars represent stable CSCIs without Significant Breach while the yellow ones show ones that did not have a Critical Breach.

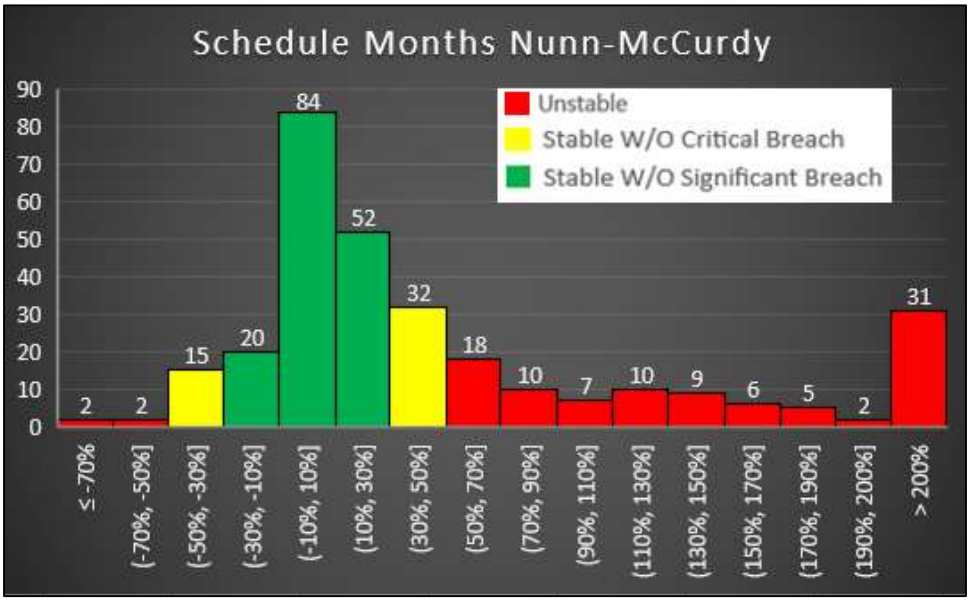


Figure 21. Schedule-Months Nunn-McCurdy Thresholds

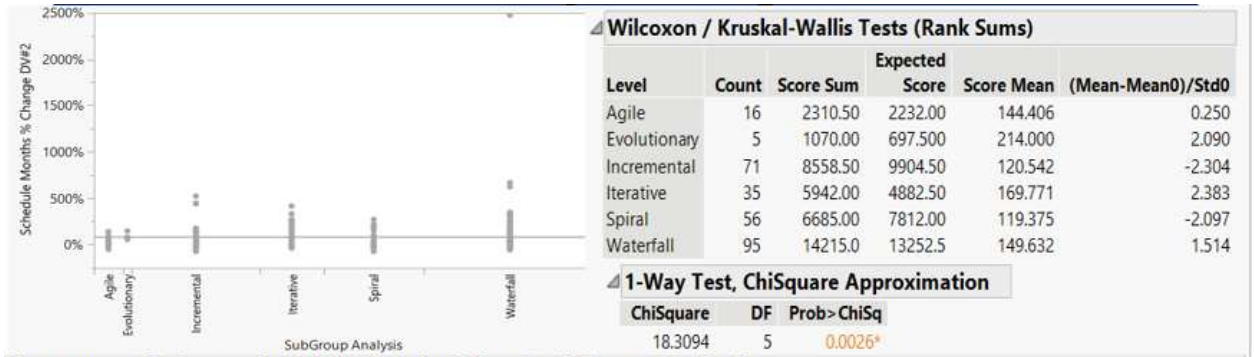
From the figures and tables above, it can be seen that cost and schedule stability in DoD software intensive programs at the CSCI level does exist, but there is a large room for improvement in the percentage of CSCIs that are considered stable. The most comparable prior definition of stability is CPI Range Stability used by Christensen and Payne. CPI Range Stability deems a program stable if the final CPI is not more than 10% +- from the initial CPI. While Christensen and Payne focused on *when* a majority of programs became stable, they also included the percent of stable programs at their initial reporting

stage, exactly like the initial reports used in this research. Their research showed 53% of programs exhibiting range stability at the 0% completion mark (Christensen and Payne, 1992). For Man-Hours, 60/335 or 17.9% of the CSCIs showed stability following the same Range Stability Rule. For Schedule-Months, the ratio was 84/305 or 27.5%. For both cost and schedule, software programs at the CSCI level show much lower levels of stability. The analysis conducted for research questions two and three help identify potential avenues to reducing the number of CSCIs finishing as unstable.

Plan Driven and Agile methodologies started seeing prevalent use in the military at different times throughout history. When analyzing the impact of different software developmental methodologies on cost and schedule stability, the newer methodology, agile in this case, would be expected to show some significant improvement on its' predecessor. This was not the case, meaning that the benefits to using Agile methodology may lie in the intangibles such as customer satisfaction rather than cost or schedule. It is also possible Agile is still too new to the DoD, meaning project managers are having difficulty accurately predicting cost and schedule for projects that use Agile. Even though Agile may show comparable instability as Plan Driven methodologies currently, Agile may still be cheaper or faster than Plan Driven methodologies. For example, a CSCI using Agile that is 50% over a $100M budget would have the same instability as a CSCI using a Plan Driven model that is 50% over a $200M budget, but the Agile CSCI would cost half as much as the alternative. Further research would be needed to determine the potential benefits or costs of Agile over its' counterparts.

While Agile and Plan Driven methodologies showed similarities, statistically significant differences were seen in the Plan Driven subgroup in terms of Schedule

Stability. Figure 22 below shows four highlighted p-values where these differences occur. These p-values suggest that there is a potential benefit of using certain Plan-Driven methodologies over others when schedule is the sole concern. The first p-value shows it is statistically more likely to finish on schedule when using Incremental rather than Iterative. The second p-value shows it is statistically more likely to finish on schedule when using Agile than Evolutionary. This was the only relationship to show significance with the Agile variable at the subgroup level. The last two p-scores show it is statistically more likely to finish on schedule when using Spiral rather than Iterative or Evolutionary.



Wilcoxon / Kruskal-Wallis Tests (Rank Sums)

Level	Count	Score Sum	Expected Score	Score Mean	(Mean-Mean0)/Std0
Agile	16	2310.50	2232.00	144.406	0.250
Evolutionary	5	1070.00	697.500	214.000	2.090
Incremental	71	8558.50	9904.50	120.542	-2.304
Iterative	35	5942.00	4882.50	169.771	2.383
Spiral	56	6685.00	7812.00	119.375	-2.097
Waterfall	95	14215.0	13252.5	149.632	1.514

1-Way Test, ChiSquare Approximation

ChiSquare	DF	Prob>ChiSq
18.3094	5	0.0026*

Nonparametric Comparisons For All Pairs Using Steel-Dwass Method

q*	Alpha
2.58852	0.1

Level	- Level	Score Mean Difference	Std Err Dif	Z	p-Value	Hodges-Lehmann	Lower CL	Upper CL	Difference Plot
Waterfall	Incremental	17.5475	7.53504	2.32879	0.1824	0.136777	-0.01899	0.40171	
Iterative	Incremental	17.2330	6.33379	2.72080	0.0712	0.520312	9.992e-16	1.42267	
Waterfall	Spiral	16.2495	7.36518	2.20627	0.2347	0.159636	-0.03090	0.42171	
Evolutionary	Agile	8.2688	3.16869	2.60951	0.0949	0.707801	-0.06828	1.43828	
Iterative	Agile	7.3313	4.48446	1.63481	0.5753	0.524257	-0.26109	1.63372	
Waterfall	Agile	1.7891	8.69644	0.20573	0.9999	0.033482	-0.33846	0.75102	
Spiral	Incremental	0.0319	6.54611	0.00488	1.0000	0.000000	-0.16601	0.13360	
Iterative	Evolutionary	-0.9143	5.58779	-0.16362	1.0000	-0.070478	-1.32641	1.79394	
Incremental	Agile	-8.0031	6.96332	-1.14932	0.8606	-0.130233	-0.42817	0.24250	
Spiral	Agile	-8.9196	5.91781	-1.50725	0.6596	-0.170500	-0.42817	0.13953	
Waterfall	Iterative	-9.4617	7.44807	-1.27035	0.8013	-0.186842	-1.02295	0.20991	
Spiral	Iterative	-15.5768	5.68268	-2.74110	0.0674	-0.543944	-1.46659	-0.00517	
Waterfall	Evolutionary	-22.3158	13.30863	-1.67679	0.5472	-0.573590	-1.38064	0.99230	
Spiral	Evolutionary	-23.4196	8.26043	-2.83516	0.0521	-0.808849	-1.47112	-0.21740	
Incremental	Evolutionary	-25.9042	10.16928	-2.54730	0.1107	-0.793369	-1.47112	0.04225	

Figure 22. Schedule-Months Subgroup Output

For the third and final research question, the impact of independent variables from the dataset on the dependent variables, Man-Hours, and Schedule-Months, was tested. A contingency table analysis was used to test the relationships between these variables. The

independent variables were determined by looking at Critical Success Factor (CSF) literature in the civilian, military, and software realms as well as looking at what variables from the dataset would potentially be changeable by the decision makers for SRDR projects. Prior literature suggests clear project goals and requirements were essential to a project being successful (Nasir and Sahibuddin, 2011). This was not supported by the results of this study, with a caveat. Results showed CSCIs were *less* likely to finish on schedule with low requirement volatility. Requirement volatility did not have any relation with staying on budget. Prior literature also suggested that support from top management was as critical, if not more so, than clear requirements. This means the impacts of poorly defined requirements could have been mitigated by upper leadership being more involved with getting the project done on time. Effective and experienced project managers and team members were also seen as a critical factor in a successful project (Nasir and Sahibuddin, 2011). These finding are partially supported by the results of this study. With a less experienced team, a project is less likely to finish on time, but more likely to finish without a breach in Man Hours. With an average experienced team, a project is less likely to finish within the initial Man Hours estimate, but more likely to finish on time. This means that the person in charge of each project must decide whether finishing within their initial Man Hours or Schedule Months estimate is more important when hiring their team and hire accordingly. The relationships between these variables as well as the variables pulled from the data rather than the literature are shown in Table 18 and summarized in the discussion below.

Table 19. Color Coded Contingency Table Analysis Results

Variable	Man-Hours Significant Breach	Man-Hours Critical Breach	Schedule-Months Significant Breach	Schedule-Months Critical Breach
New/Upgrade				
Req Volatility Initial 0				
Req Volatility Initial 1				
Req Volatility Initial 2				
Req Volatility Initial 3				
Req Volatility Initial 4				
Req Volatility Initial 5				
Req Volatility Filtered 0				
Req Volatility Filtered 1				
Req Volatility Filtered 2				
Req Volatility Filtered 3				
Req Volatility Filtered 4				
Req Volatility Filtered 5				
Contractor Bae Systems				
Contractor Boeing				
Contractor General Dynamics				
Contractor Lockheed Martin				

Contractor Northrop Grumman				
Contractor Raytheon				
Team Experience Level Low				
Team Experience Level Average				
Team Experience Level High				
Service Air Force				
Service Army				
Service Navy				
Programming Language Ada				
Programming Language C				
Programming Language C#				
Programming Language C++				
Programming Language C/C++				
Programming Language Java				

Table Legend:
Yellow = Negative Correlation, p-value < 0.10
Orange = Negative Correlation, p-value < 0.05
Red = Negative Correlation, p-value < 0.01
Light Blue = Positive Correlation, p-value < 0.10
Light Green = Positive Correlation, p-value < 0.05
Dark Green = Positive Correlation, p-value < 0.01

Relationships with the Man-Hours variable suggests that SRDR CSCIs are *less* likely to finish on budget with General Dynamics as the lead contractor, with an average

experience level team, programmed in C, or if managed by the Army. SRDR are *more* likely to finish on budget if Lockheed Martin is the lead contractor or the team has a low level of average experience. Relationships with the Schedule-Months variable suggest that SRDR CSCIs are *less* likely to finish on schedule if the team has a low level of average experience, if requirement volatility is low, if Northrop Grumman is the lead contractor, managed by Navy, or programmed in C. SRDR CSCIs are *more* likely to finish on schedule if requirement volatility is above average, a team has average level of average experience, Lockheed Martin or Boeing is the lead contractor, managed by Army, or programmed in Ada. All other tested variables showed in insignificant relationship with the Man-Hours and Schedule-Months variables.

Limitations

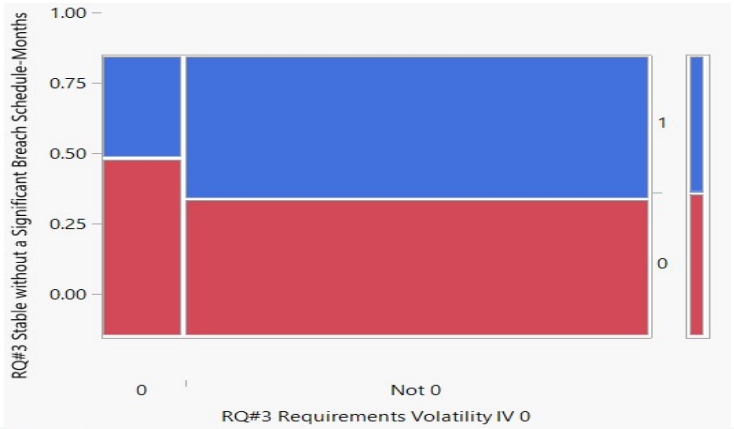
The biggest limitation for this research stemmed from the lack of EVM data available for SRDR programs, resulting in a shift in the way the research was conducted. Even when shifting to SRDR data, of the 4499 CSCIs collected from CADE on SRDR programs, only 335 CSCIs were usable for the initial research with further reductions from there. These exclusions were due to lack of complete data and reporting inconsistencies by contractors. Furthermore, findings may be limited due to limited sample size of CSCIs using Agile processes. Findings may also be limited due to potential spurious relationships between variables with p-values between 0.1 and 0.01.Lastly, bivariate contingency tables may result in some omitted variable biases. While this was addressed with the multivariate analysis looking at correlation between size and the examined variables, this potentially bias cannot be completely ignored.

Final Thoughts

This research expanded the knowledge on stability in DoD programs while setting the groundwork on stability in SRDR specific programs at the CSCI level. Some variables like Service and New/Upgrade cannot be controlled by the project manager, but others are more malleable. Programming Language, Requirement Volatility, Contractor, and Team Experience Level are all able to be adjusted to some extent before or at the beginning of a project. The importance of giving each project the best chances of finishing on time and within budget cannot be overstated. The importance of further research into SRDR CSCIs is crucial based on the current lack of existing literature. If program offices can better grasp the program's cost and schedule drivers at the CSCI level, stability at the program will also improve, saving the DoD both time and money.

Appendix – Contingency Table Analysis Results

A contingency table analysis is used to study relationships between variables, identified when Pearson's chi-squared test is significant at a p-value of less than 0.10. This Appendix includes all significant contingency table tests for all independent variables tested.



Contingency Table

RQ#3 Stable without a Significant Breach Schedule-Months

RQ#3 Requirements Volatility IV 0	Count / Expected	0	1	Total
0		26	15	41
		20.9556	20.0444	
Not 0		112	117	229
		117.044	111.956	
Total		138	132	270

Tests

N	DF	-LogLike	RSquare (U)
270	1	1.4817600	0.0079

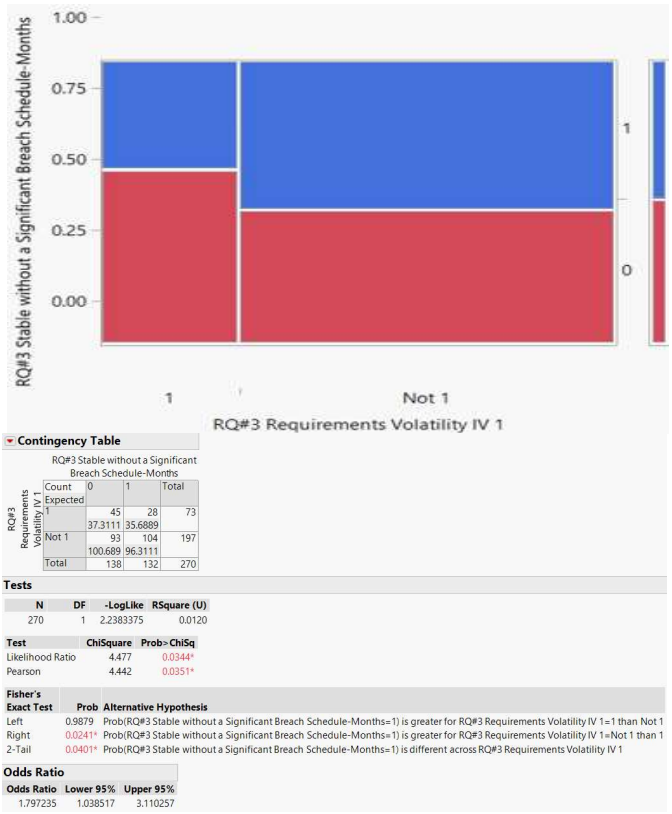
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	2.964	0.0852
Pearson	2.929	0.0870

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.9706	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is greater for RQ#3 Requirements Volatility IV 0=0 than Not 0
Right	0.0611	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is greater for RQ#3 Requirements Volatility IV 0=Not 0 than 0
2-Tail	0.0928	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is different across RQ#3 Requirements Volatility IV 0

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
1.810714	0.911598	3.596637

Significant Breach Schedule-Months by Requirements Volatility Initial 0



Contingency Table

RQ#3 Stable without a Significant Breach Schedule-Months

RQ#3 Requirements Volatility IV 1	Count / Expected	0	1	Total
1		45	28	73
		37.3111	35.6889	
Not 1		93	104	197
		100.689	96.3111	
Total		138	132	270

Tests

N	DF	-LogLike	RSquare (U)
270	1	2.2383375	0.0120

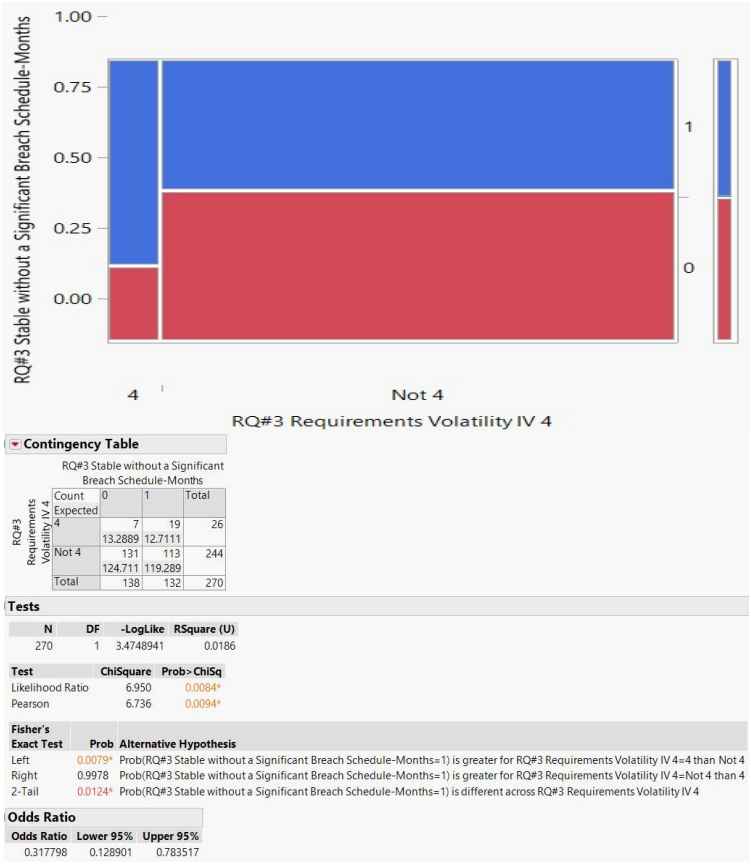
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	4.477	0.0344*
Pearson	4.442	0.0351*

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.9879	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is greater for RQ#3 Requirements Volatility IV 1=1 than Not 1
Right	0.0241*	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is greater for RQ#3 Requirements Volatility IV 1=Not 1 than 1
2-Tail	0.0401*	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is different across RQ#3 Requirements Volatility IV 1

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
1.797235	1.038517	3.110257

Significant Breach Schedule-Months by Requirements Volatility Initial 1



Contingency Table

RQ#3 Requirements Volatility IV 4	Count / Expected	RQ#3 Stable without a Significant Breach Schedule-Months: 0	1	Total
4	Count	7	19	26
	Expected	13.2889	12.7111	
Not 4	Count	131	113	244
	Expected	124.711	119.289	
Total		138	132	270

Tests

N	DF	-LogLike	RSquare (U)
270	1	3.4748941	0.0186

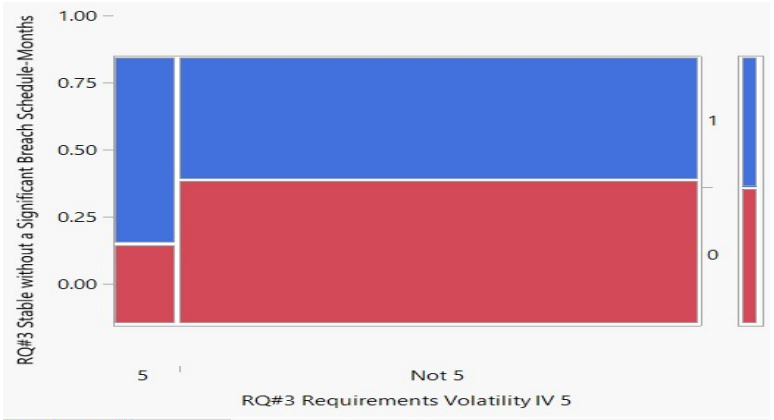
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	6.950	0.0084*
Pearson	6.736	0.0094*

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.0079*	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is greater for RQ#3 Requirements Volatility IV 4=4 than Not 4
Right	0.9978	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is greater for RQ#3 Requirements Volatility IV 4=Not 4 than 4
2-Tail	0.0124*	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is different across RQ#3 Requirements Volatility IV 4

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
0.317798	0.128901	0.783517

Significant Breach Schedule-Months by Requirements Volatility Initial 4



Contingency Table

RQ#3 Stable without a Significant Breach Schedule-Months

RQ#3 Requirements Volatility IV 5	Count / Expected	0	1	Total
5		9	21	30
		15.3333	14.6667	
Not 5		129	111	240
		122.667	117.333	
Total		138	132	270

Tests

N	DF	-LogLike	RSquare (U)
270	1	3.0774484	0.0164

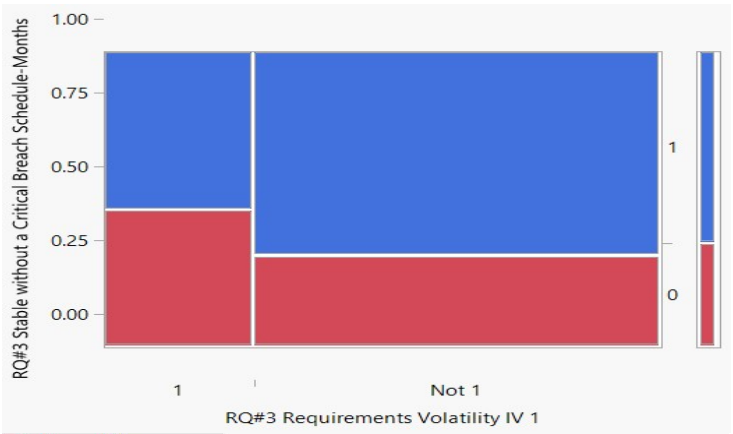
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	6.155	0.0131*
Pearson	6.020	0.0141*

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.0114*	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is greater for RQ#3 Requirements Volatility IV 5=5 than Not 5
Right	0.9963	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is greater for RQ#3 Requirements Volatility IV 5=Not 5 than 5
2-Tail	0.0192*	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is different across RQ#3 Requirements Volatility IV 5

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
0.368771	0.162246	0.838183

Significant Breach Schedule-Months by Requirements Volatility Initial 5



Contingency Table

RQ#3 Stable without a Critical Breach Schedule-Months

RQ#3 Requirements Volatility IV 1	Count / Expected	0	1	Total
1		34	39	73
		25.6852	47.3148	
Not 1		61	136	197
		69.3148	127.685	
Total		95	175	270

Tests

N	DF	-LogLike	RSquare (U)
270	1	2.7829372	0.0159

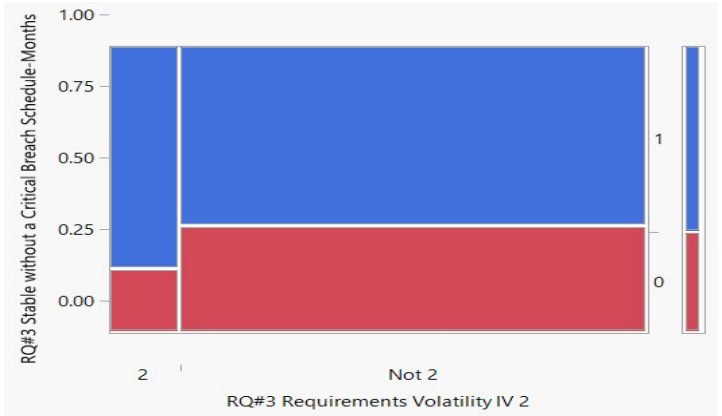
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	5.566	0.0183*
Pearson	5.692	0.0170*

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.9939	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is greater for RQ#3 Requirements Volatility IV 1=1 than Not 1
Right	0.0131*	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is greater for RQ#3 Requirements Volatility IV 1=Not 1 than 1
2-Tail	0.0216*	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is different across RQ#3 Requirements Volatility IV 1

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
1.943674	1.121195	3.369503

Critical Breach Schedule-Months by Requirements Volatility Initial 1



Contingency Table

RQ#3 Stable without a Critical Breach Schedule-Months

RQ#3 Requirements Volatility IV 2	Count / Expected	0	1	Total
2		8	28	36
		12.6667	23.3333	
Not 2		87	147	234
		82.3333	151.667	
Total		95	175	270

Tests

N	DF	-LogLike	RSquare (U)
270	1	1.6311112	0.0093

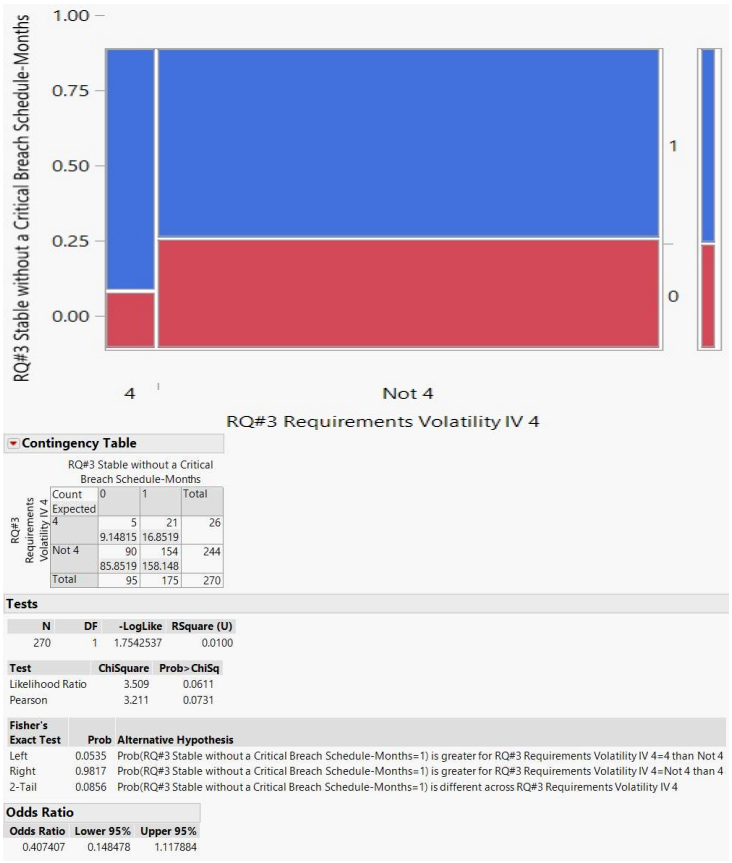
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	3.262	0.0709
Pearson	3.061	0.0802

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.0561	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is greater for RQ#3 Requirements Volatility IV 2=2 than Not 2
Right	0.9767	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is greater for RQ#3 Requirements Volatility IV 2=Not 2 than 2
2-Tail	0.0929	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is different across RQ#3 Requirements Volatility IV 2

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
0.482759	0.210663	1.106295

Critical Breach Schedule-Months by Requirements Volatility Initial 2



Contingency Table

RQ#3 Stable without a Critical Breach Schedule-Months

RQ#3 Requirements Volatility IV 4	Count / Expected	0	1	Total
4		5	21	26
		9.14815	16.8519	
Not 4		90	154	244
		85.8519	158.148	
Total		95	175	270

Tests

N	DF	-LogLike	RSquare (U)
270	1	1.7542537	0.0100

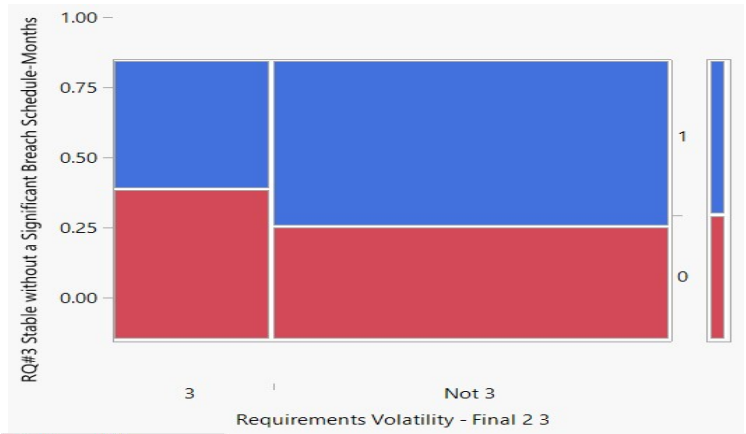
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	3.509	0.0611
Pearson	3.211	0.0731

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.0535	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is greater for RQ#3 Requirements Volatility IV 4=4 than Not 4
Right	0.9817	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is greater for RQ#3 Requirements Volatility IV 4=Not 4 than 4
2-Tail	0.0856	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is different across RQ#3 Requirements Volatility IV 4

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
0.407407	0.148478	1.117884

Critical Breach Schedule-Months by Requirements Volatility Initial 4



Contingency Table

RQ#3 Stable without a Significant Breach Schedule-Months

Requirements Volatility - Final 2 3	Count / Expected	0	1	Total
3		33	28	61
		27.2066	33.7934	
Not 3		62	90	152
		67.7934	84.2066	
Total		95	118	213

Tests

N	DF	-LogLike	RSquare (U)
213	1	1.5546740	0.0106

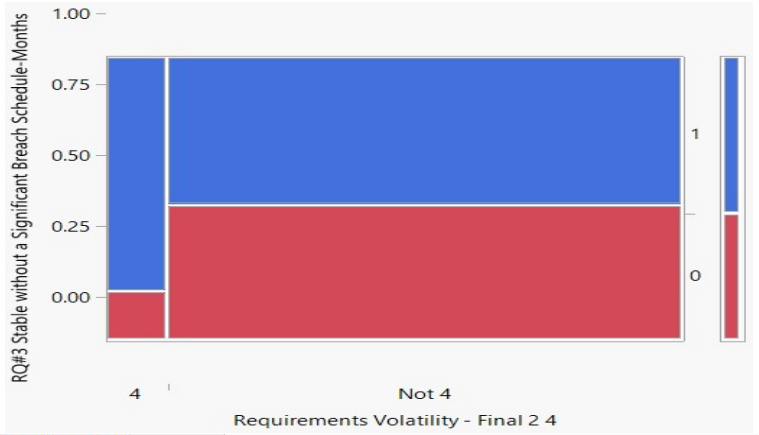
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	3.109	0.0778
Pearson	3.121	0.0773

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.9723	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is greater for Requirements Volatility - Final 2 3=3 than Not 3
Right	0.0535	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is greater for Requirements Volatility - Final 2 3=Not 3 than 3
2-Tail	0.0937	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is different across Requirements Volatility - Final 2 3

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
1.710829	0.940299	3.112772

Significant Breach Schedule-Months by Requirements Volatility Filtered 3



Contingency Table

RQ#3 Stable without a Significant Breach Schedule-Months

Requirements Volatility - Final 2 4	Count / Expected	0	1	Total
4		4	19	23
		10.2582	12.7418	
Not 4		91	99	190
		84.7418	105.258	
Total		95	118	213

Tests

N	DF	-LogLike	RSquare (U)
213	1	4.2397985	0.0290

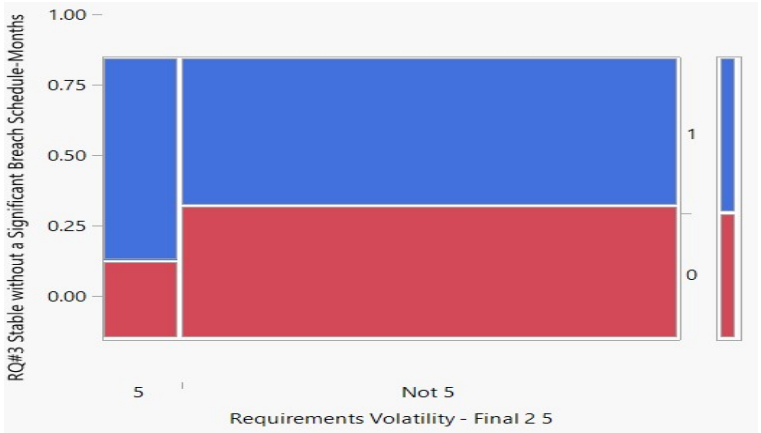
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	8.480	0.0036*
Pearson	7.726	0.0054*

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.0041*	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is greater for Requirements Volatility - Final 2 4=4 than Not 4
Right	0.9992	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is greater for Requirements Volatility - Final 2 4=Not 4 than 4
2-Tail	0.0068*	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is different across Requirements Volatility - Final 2 4

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
0.229034	0.075092	0.698561

Significant Breach Schedule-Months by Requirements Volatility Filtered 4



Contingency Table

RQ#3 Stable without a Significant Breach Schedule-Months

Requirements Volatility - Final 2 5	Count / Expected	0	1	Total
5		8	21	29
		12.9343	16.0657	
Not 5		87	97	184
		82.0657	101.934	
Total		95	118	213

Tests

N	DF	-LogLike	RSquare (U)
213	1	2.0478572	0.0140

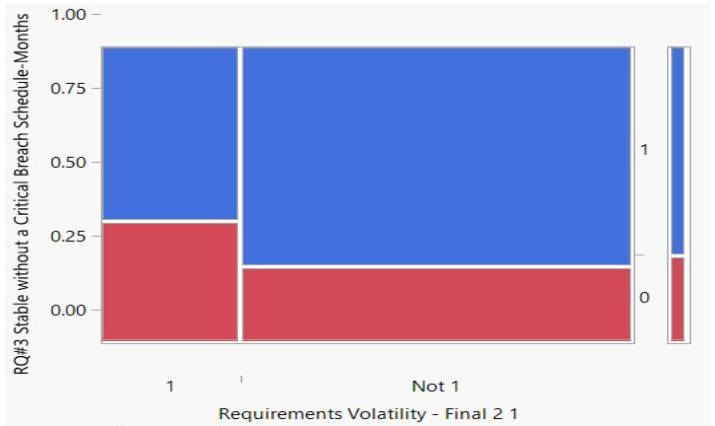
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	4.096	0.0430*
Pearson	3.933	0.0473*

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.0357*	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is greater for Requirements Volatility - Final 2 5=5 than Not 5
Right	0.9870	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is greater for Requirements Volatility - Final 2 5=Not 5 than 5
2-Tail	0.0692	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is different across Requirements Volatility - Final 2 5

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
0.42474	0.178978	1.007968

Significant Breach Schedule-Months by Requirements Volatility Filtered 5



Contingency Table

RQ#3 Stable without a Critical Breach Schedule-Months

Requirements Volatility - Final 2 1	Count / Expected	0	1	Total
1		23	33	56
		16.5634	39.4366	
Not 1		40	117	157
		46.4366	110.563	
Total		63	150	213

Tests

N	DF	-LogLike	RSquare (U)
213	1	2.3228083	0.0180

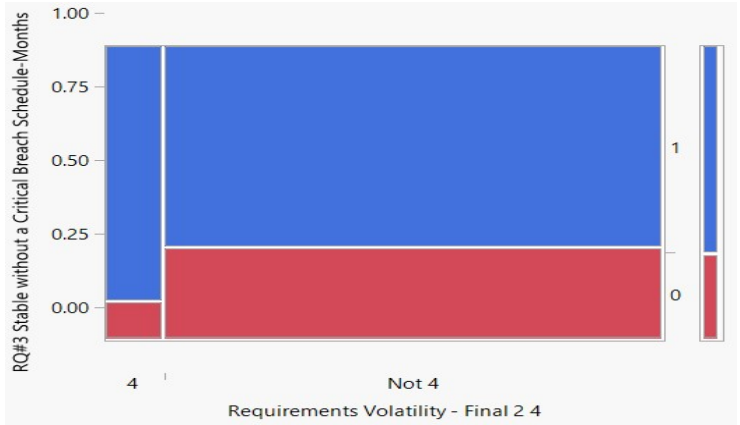
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	4.646	0.0311*
Pearson	4.819	0.0282*

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.9901	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is greater for Requirements Volatility - Final 2 1=1 than Not 1
Right	0.0228*	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is greater for Requirements Volatility - Final 2 1=Not 1 than 1
2-Tail	0.0400*	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is different across Requirements Volatility - Final 2 1

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
2.038636	1.072698	3.874377

Critical Breach Schedule-Months by Requirements Volatility Filtered 1



Contingency Table

RQ#3 Stable without a Critical Breach Schedule-Months

Requirements Volatility - Final 2 4	Count / Expected	0	1	Total
4		3	20	23
		6.80282	16.1972	
Not 4		60	130	190
		56.1972	133.803	
Total		63	150	213

Tests

N	DF	-LogLike	RSquare (U)
213	1	1.9421434	0.0150

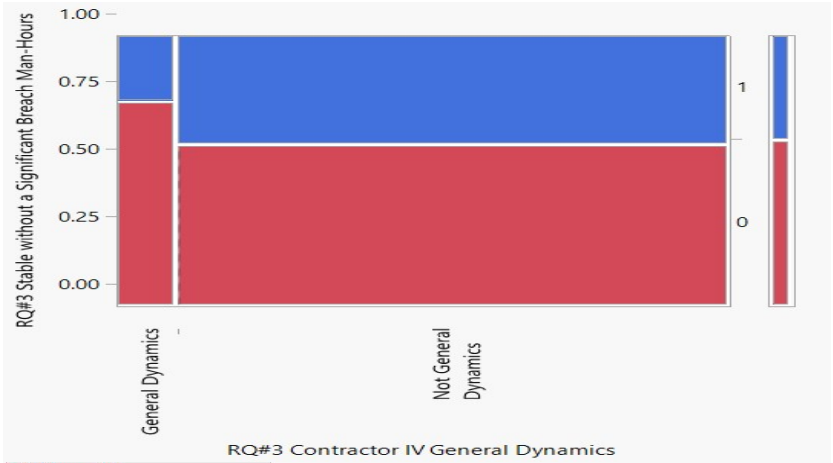
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	3.884	0.0487*
Pearson	3.384	0.0658

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.0489*	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is greater for Requirements Volatility - Final 2 4=4 than Not 4
Right	0.9869	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is greater for Requirements Volatility - Final 2 4=Not 4 than 4
2-Tail	0.0891	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is different across Requirements Volatility - Final 2 4

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
0.325	0.092979	1.13601

Critical Breach Schedule-Months by Requirements Volatility Filtered 4



Contingency Table

RQ#3 Contractor IV General Dynamics	Count / Expected	0	1	Total
General Dynamics	Count	25	8	33
	Expected	20.2925	12.7075	
Not General Dynamics	Count	181	121	302
	Expected	185.707	116.293	
Total		206	129	335

RQ#3 Stable without a Significant Breach Man-Hours

Tests

N	DF	-LogLike	RSquare (U)
335	1	1.6677751	0.0075

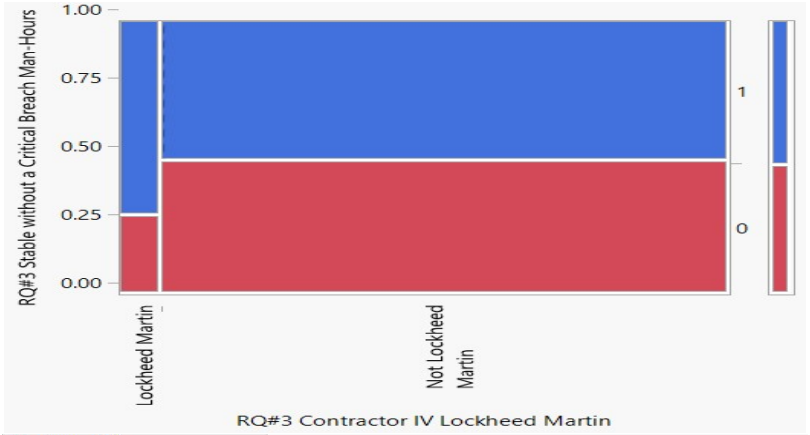
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	3.336	0.0678
Pearson	3.146	0.0761

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.9779	Prob(RQ#3 Stable without a Significant Breach Man-Hours=1) is greater for RQ#3 Contractor IV General Dynamics=General Dynamics than Not General Dynamics
Right	0.0537	Prob(RQ#3 Stable without a Significant Breach Man-Hours=1) is greater for RQ#3 Contractor IV General Dynamics=Not General Dynamics than General Dynamics
2-Tail	0.0906	Prob(RQ#3 Stable without a Significant Breach Man-Hours=1) is different across RQ#3 Contractor IV General Dynamics

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
2.089088	0.912093	4.784918

Significant Breach Man-Hours by Contractor General Dynamics



Contingency Table

RQ#3 Stable without a Critical Breach Man-Hours

RQ#3 Contractor IV Lockheed Martin	Count / Expected	0	1	Total
Lockheed Martin	Count	7	17	24
	Expected	11.391	12.609	
Not Lockheed Martin	Count	152	159	311
	Expected	147.609	163.391	
Total		159	176	335

Tests

N	DF	-LogLike	RSquare (U)
335	1	1.7954945	0.0077

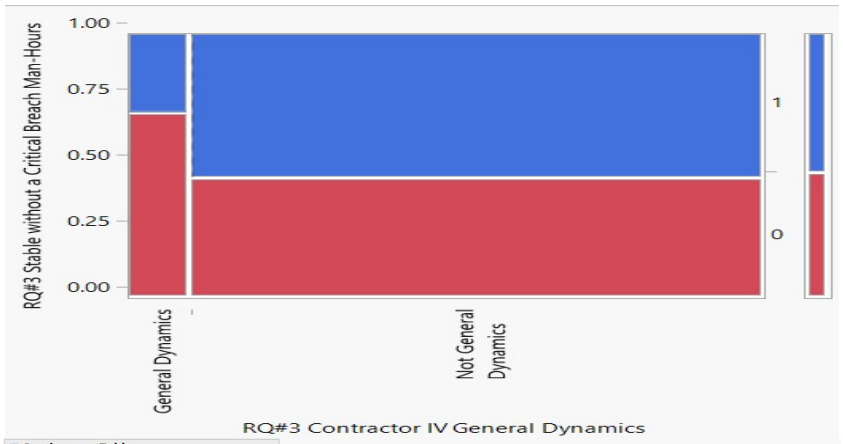
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	3.591	0.0581
Pearson	3.470	0.0625

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.0479*	Prob(RQ#3 Stable without a Critical Breach Man-Hours=1) is greater for RQ#3 Contractor IV Lockheed Martin=Lockheed Martin than Not Lockheed Martin
Right	0.9825	Prob(RQ#3 Stable without a Critical Breach Man-Hours=1) is greater for RQ#3 Contractor IV Lockheed Martin=Not Lockheed Martin than Lockheed Martin
2-Tail	0.0882	Prob(RQ#3 Stable without a Critical Breach Man-Hours=1) is different across RQ#3 Contractor IV Lockheed Martin

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
0.430728	0.173752	1.067762

Critical Breach Man-Hours by Contractor Lockheed Martin



Contingency Table

RQ#3 Contractor IV General Dynamics	RQ#3 Stable without a Critical ...	0	1	Total
	Count			
	Expected			
	General Dynamics	23	10	33
		15.6627	17.3373	
	Not General Dynamics	136	166	302
		143.337	158.663	
	Total	159	176	335

Tests

N	DF	-LogLike	RSquare (U)
335	1	3.6923233	0.0159

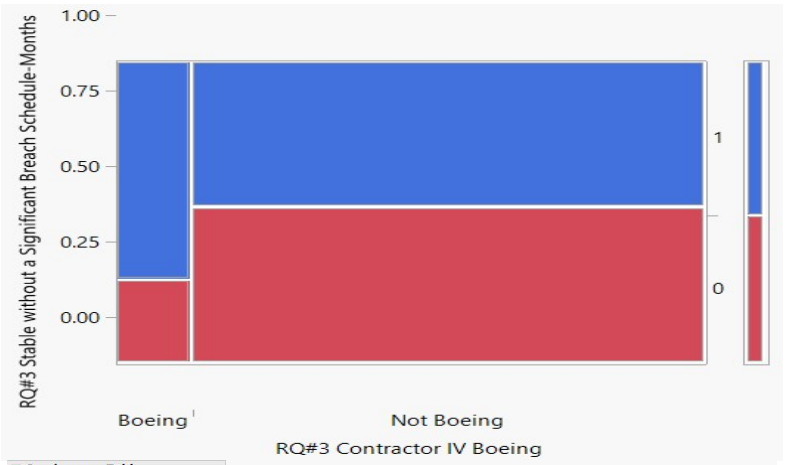
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	7.385	0.0066*
Pearson	7.257	0.0071*

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.9982	Prob(RQ#3 Stable without a Critical Breach Man-Hours=1) is greater for RQ#3 Contractor IV General Dynamics=General Dynamics than Not General Dynamics
Right	0.0058*	Prob(RQ#3 Stable without a Critical Breach Man-Hours=1) is greater for RQ#3 Contractor IV General Dynamics=Not General Dynamics than General Dynamics
2-Tail	0.0095*	Prob(RQ#3 Stable without a Critical Breach Man-Hours=1) is different across RQ#3 Contractor IV General Dynamics

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
2.807353	1.291751	6.101197

Critical Breach Man-Hours by Contractor General Dynamics



Contingency Table

RQ#3 Stable without a Significant Breach Schedule-Months

RQ#3 Contractor IV Boeing	Count / Expected	0	1	Total
Boeing	Count	11	29	40
	Expected	19.541	20.459	
Not Boeing	Count	138	127	265
	Expected	129.459	135.541	
Total		149	156	305

Tests

N	DF	-LogLike	RSquare (U)
305	1	4.3471687	0.0206

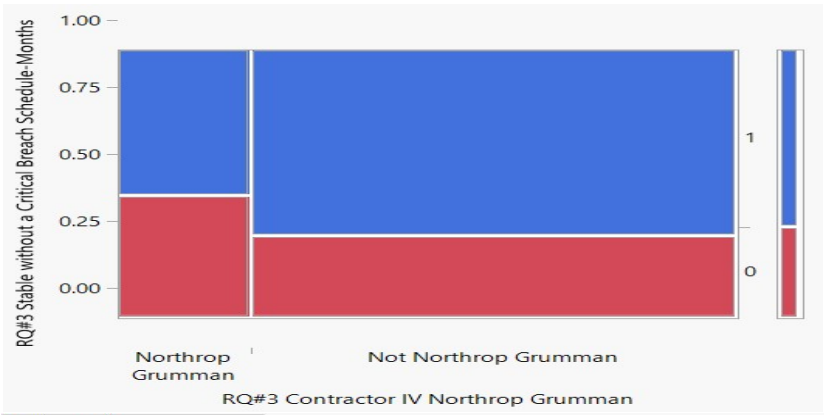
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	8.694	0.0032*
Pearson	8.400	0.0038*

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.0028*	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is greater for RQ#3 Contractor IV Boeing=Boeing than Not Boeing
Right	0.9991	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is greater for RQ#3 Contractor IV Boeing=Not Boeing than Boeing
2-Tail	0.0039*	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is different across RQ#3 Contractor IV Boeing

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
0.349075	0.167435	0.727766

Significant Breach Schedule-Months by Contractor Boeing



Contingency Table

RQ#3 Stable without a Critical Breach Schedule-Months

RQ#3 Contractor IV Northrop Grumman	Count / Expected	0	1	Total
Northrop Grumman	Count	30	36	66
	Expected	22.2885	43.7115	
Not Northrop Grumman	Count	73	166	239
	Expected	80.7115	158.289	
Total		103	202	305

Tests

N	DF	-LogLike	RSquare (U)
305	1	2.4920446	0.0128

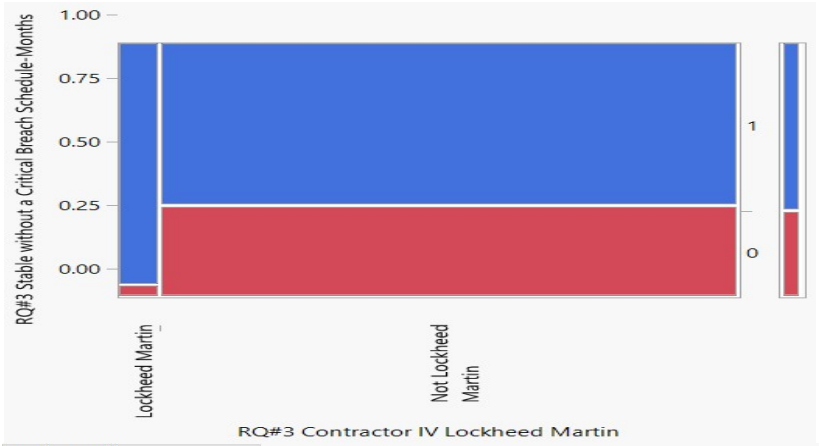
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	4.984	0.0256*
Pearson	5.141	0.0234*

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.9914	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is greater for RQ#3 Contractor IV Northrop Grumman=Northrop Grumman than Not Northrop Grumman
Right	0.0180*	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is greater for RQ#3 Contractor IV Northrop Grumman=Not Northrop Grumman than Northrop Grumman
2-Tail	0.0276*	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is different across RQ#3 Contractor IV Northrop Grumman

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
1.894977	1.085416	3.308351

Critical Breach Schedule-Months by Contractor Northrop Grumman



Contingency Table

RQ#3 Stable without a Critical Breach Schedule-Months

RQ#3 Contractor IV Lockheed Martin		0	1	Total
	Count			
	Expected			
	Lockheed Martin	1	20	21
		7.0918	13.9082	
	Not Lockheed Martin	102	182	284
		95.9082	188.092	
	Total	103	202	305

Tests

N	DF	-LogLike	RSquare (U)
305	1	5.5953658	0.0287

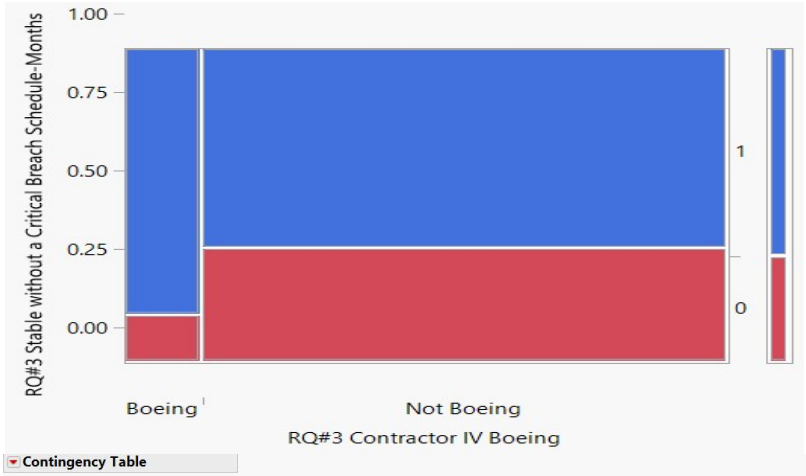
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	11.191	0.0008*
Pearson	8.485	0.0036*

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.0016*	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is greater for RQ#3 Contractor IV Lockheed Martin=Lockheed Martin than Not Lockheed Martin
Right	0.9999	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is greater for RQ#3 Contractor IV Lockheed Martin=Not Lockheed Martin than Lockheed Martin
2-Tail	0.0030*	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is different across RQ#3 Contractor IV Lockheed Martin

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
0.089216	0.0118	0.674519

Critical Breach Schedule-Months by Contractor Lockheed Martin



Contingency Table

RQ#3 Stable without a Critical Breach Schedule-Months

RQ#3 Contractor IV Boeing	Count / Expected	0	1	Total
Boeing		6	34	40
		13.5082	26.4918	
Not Boeing		97	168	265
		89.4918	175.508	
Total		103	202	305

Tests

N	DF	-LogLike	RSquare (U)
305	1	4.0840773	0.0209

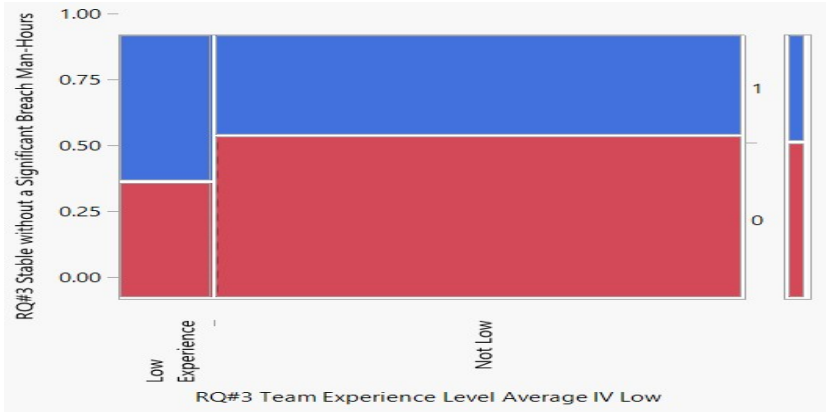
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	8.168	0.0043*
Pearson	7.252	0.0071*

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.0043*	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is greater for RQ#3 Contractor IV Boeing=Boeing than Not Boeing
Right	0.9988	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is greater for RQ#3 Contractor IV Boeing=Not Boeing than Boeing
2-Tail	0.0068*	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is different across RQ#3 Contractor IV Boeing

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
0.30564	0.123872	0.754131

Critical Breach Schedule-Months by Contractor Boeing



Contingency Table

RQ#3 Stable without a Significant Breach Man-Hours

RQ#3 Team Experience Level Average IV Low		0	1	Total
	Count			
	Expected			
Low Experience		19	24	43
		25.4929	17.5071	
Not Low		147	90	237
		140.507	96.4929	
Total		166	114	280

Tests

N	DF	-LogLike	RSquare (U)
280	1	2.3567276	0.0125

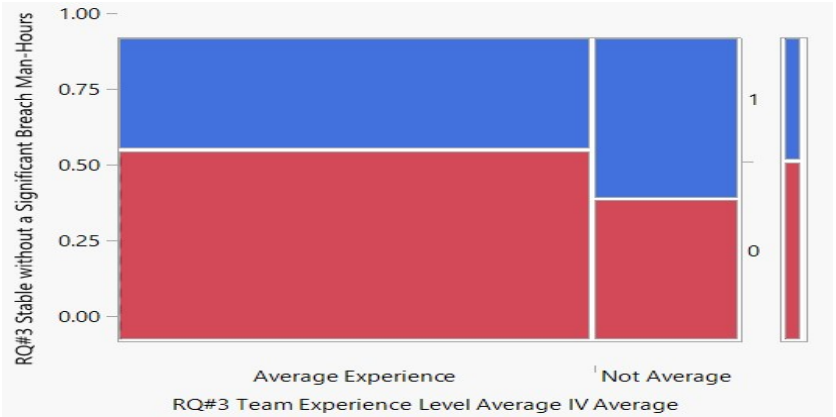
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	4.713	0.0299*
Pearson	4.799	0.0285*

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.0223*	Prob(RQ#3 Stable without a Significant Breach Man-Hours=1) is greater for RQ#3 Team Experience Level Average IV Low=Low Experience than Not Low
Right	0.9904	Prob(RQ#3 Stable without a Significant Breach Man-Hours=1) is greater for RQ#3 Team Experience Level Average IV Low=Not Low than Low Experience
2-Tail	0.0420*	Prob(RQ#3 Stable without a Significant Breach Man-Hours=1) is different across RQ#3 Team Experience Level Average IV Low

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
0.484694	0.25138	0.934552

Significant Breach Man-Hours by Team Experience Level Low



Contingency Table

RQ#3 Team Experience Level Average IV Average	Count / Expected	RQ#3 Stable without a Significant Breach Man-Hours 0	1	Total
Average Experience	Count	135	79	214
	Expected	126.871	87.1286	
Not Average	Count	31	35	66
	Expected	39.1286	26.8714	
Total		166	114	280

Tests

N	DF	-LogLike	RSquare (U)
280	1	2.6776736	0.0142

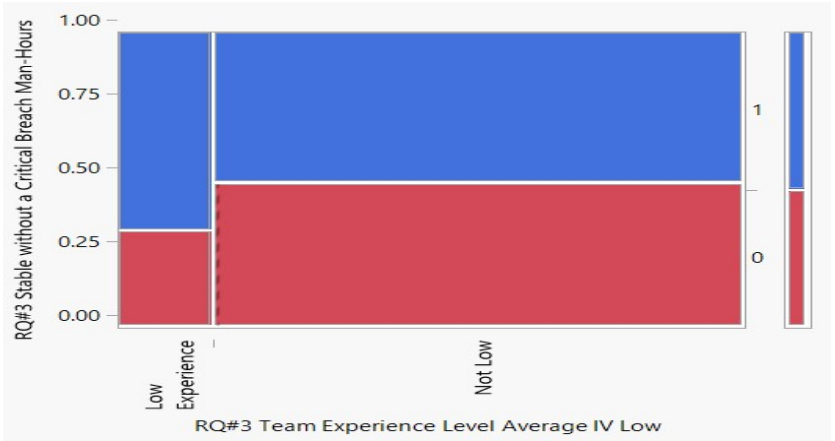
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	5.355	0.0207*
Pearson	5.427	0.0198*

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.9930	Prob(RQ#3 Stable without a Significant Breach Man-Hours=1) is greater for RQ#3 Team Experience Level Average IV Average=Average Experience than Not Average
Right	0.0148*	Prob(RQ#3 Stable without a Significant Breach Man-Hours=1) is greater for RQ#3 Team Experience Level Average IV Average=Not Average than Average Experience
2-Tail	0.0223*	Prob(RQ#3 Stable without a Significant Breach Man-Hours=1) is different across RQ#3 Team Experience Level Average IV Average

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
1.929359	1.104876	3.369088

Significant Breach Man-Hours by Team Experience Level Average



Contingency Table

RQ#3 Stable without a Critical Breach Man-Hours

RQ#3 Team Experience Level Average IV Low		0	1	Total
	Count			
	Expected			
Low Experience		14	29	43
		19.9643	23.0357	
Not Low		116	121	237
		110.036	126.964	
Total		130	150	280

Tests

N	DF	-LogLike	RSquare (U)
280	1	2.0099577	0.0104

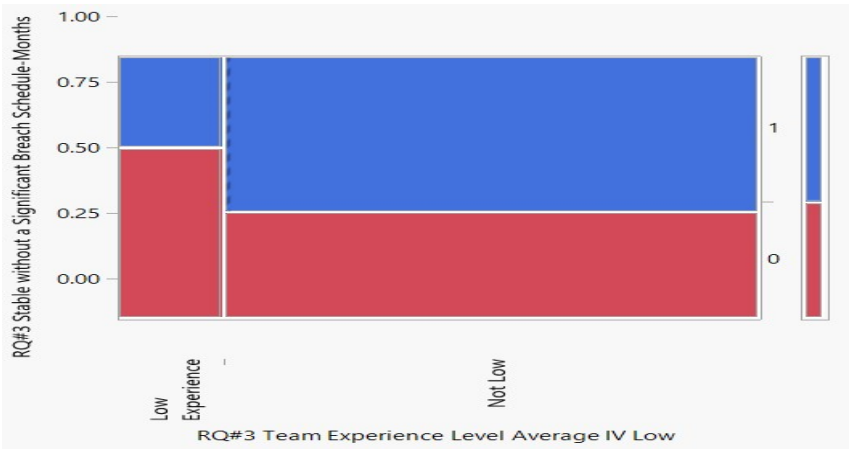
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	4.020	0.0450*
Pearson	3.930	0.0474*

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.0337*	Prob(RQ#3 Stable without a Critical Breach Man-Hours=1) is greater for RQ#3 Team Experience Level Average IV Low=Low Experience than Not Low
Right	0.9850	Prob(RQ#3 Stable without a Critical Breach Man-Hours=1) is greater for RQ#3 Team Experience Level Average IV Low=Not Low than Low Experience
2-Tail	0.0667	Prob(RQ#3 Stable without a Critical Breach Man-Hours=1) is different across RQ#3 Team Experience Level Average IV Low

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
0.503567	0.253382	1.000779

Critical Breach Man-Hours by Team Experience Level Low



Contingency Table

RQ#3 Stable without a Significant Breach Schedule-Months

RQ#3 Team Experience Level Average IV Low	Count / Expected	0	1	Total
Low Experience	Count	28	15	43
	Expected	19.1667	23.8333	
Not Low	Count	87	128	215
	Expected	95.8333	119.167	
Total		115	143	258

Tests

N	DF	-LogLike	RSquare (U)
258	1	4.4071744	0.0249

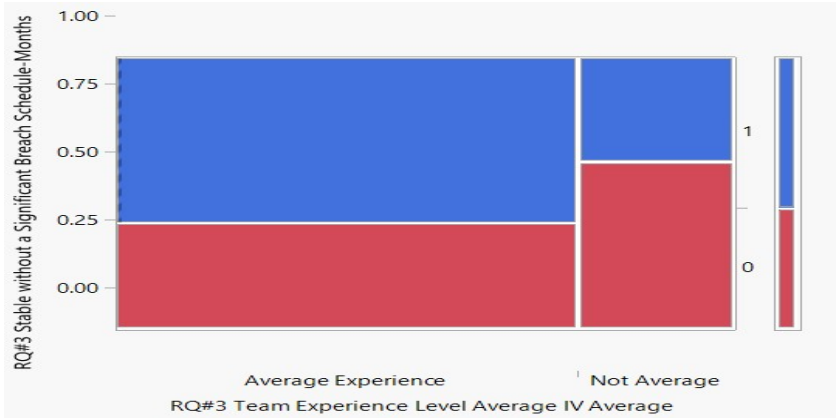
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	8.814	0.0030*
Pearson	8.814	0.0030*

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.9992	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is greater for RQ#3 Team Experience Level Average IV Low=Low Experience than Not Low
Right	0.0026*	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is greater for RQ#3 Team Experience Level Average IV Low=Not Low than Low Experience
2-Tail	0.0040*	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is different across RQ#3 Team Experience Level Average IV Low

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
2.74636	1.386201	5.441124

Significant Breach Schedule-Months by Team Experience Level Low



Contingency Table

RQ#3 Stable without a Significant Breach Schedule-Months

RQ#3 Team Experience Level Average IV Average	Count / Expected	0	1	Total
Average Experience	Count	75	118	193
	Expected	86.0271	106.973	
Not Average	Count	40	25	65
	Expected	28.9729	36.0271	
Total		115	143	258

Tests

N	DF	-LogLike	RSquare (U)
258	1	5.0546850	0.0285

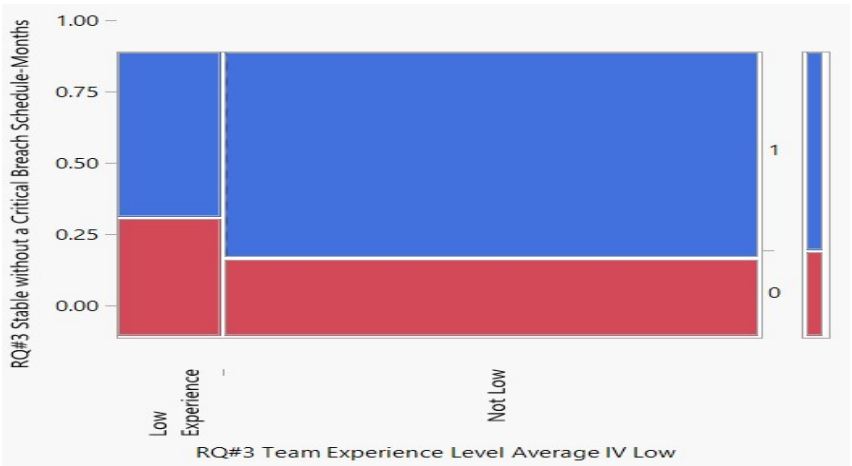
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	10.109	0.0015*
Pearson	10.122	0.0015*

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.0012*	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is greater for RQ#3 Team Experience Level Average IV Average=Average Experience than Not Average
Right	0.9996	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is greater for RQ#3 Team Experience Level Average IV Average=Not Average than Average Experience
2-Tail	0.0023*	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is different across RQ#3 Team Experience Level Average IV Average

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
0.397246	0.222981	0.707702

Significant Breach Schedule-Months by Team Experience Level Average



Contingency Table

RQ#3 Team Experience Level Average IV Low	RQ#3 Stable without a Critical Breach Schedule-Months		
	0	1	Total
Low Experience	18	25	43
	13	30	
Not Low	60	155	215
	65	150	
Total	78	180	258

Tests

N	DF	-LogLike	RSquare (U)
258	1	1.5794244	0.0100

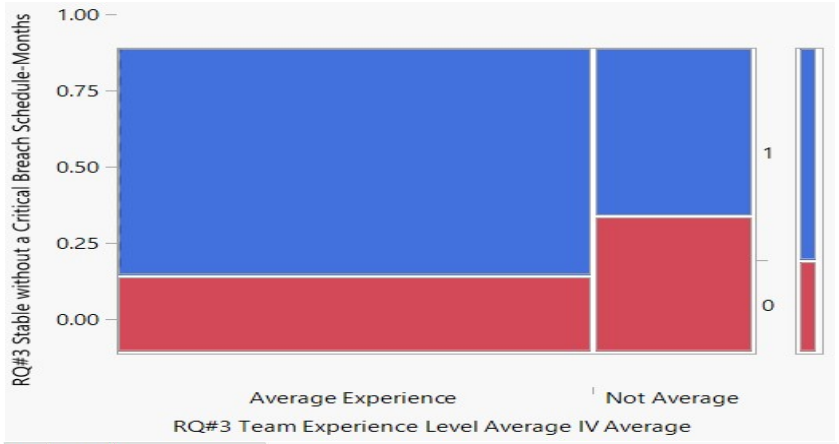
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	3.159	0.0755
Pearson	3.308	0.0690

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.9753	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is greater for RQ#3 Team Experience Level Average IV Low=Low Experience than Not Low
Right	0.0531	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is greater for RQ#3 Team Experience Level Average IV Low=Not Low than Low Experience
2-Tail	0.1004	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is different across RQ#3 Team Experience Level Average IV Low

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
1.86	0.946852	3.653793

Critical Breach Schedule-Months by Team Experience Level Low



Contingency Table

RQ#3 Stable without a Critical Breach Schedule-Months

RQ#3 Team Experience Level Average IV Average		0	1	Total
	Count			
	Expected			
Average Experience		49	144	193
		58.3488	134.651	
Not Average		29	36	65
		19.6512	45.3488	
Total		78	180	258

Tests

N	DF	-LogLike	RSquare (U)
258	1	4.0842429	0.0258

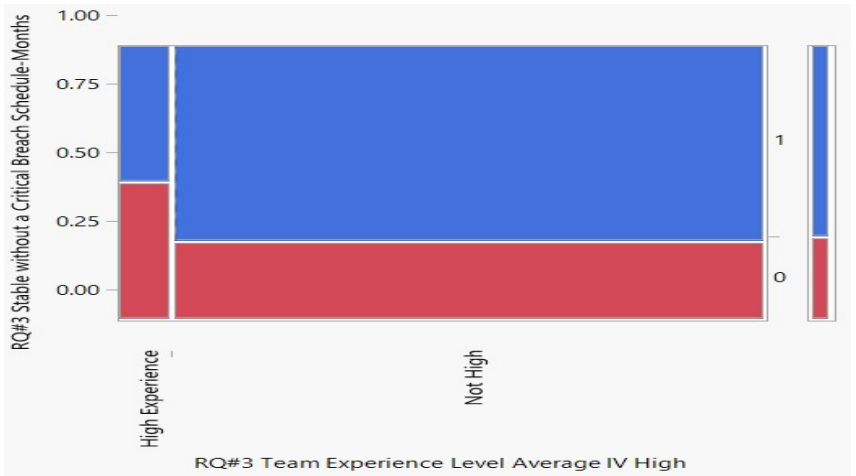
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	8.168	0.0043*
Pearson	8.522	0.0035*

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.0033*	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is greater for RQ#3 Team Experience Level Average IV Average=Average Experience than Not Average
Right	0.9987	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is greater for RQ#3 Team Experience Level Average IV Average=Not Average than Average Experience
2-Tail	0.0048*	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is different across RQ#3 Team Experience Level Average IV Average

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
0.422414	0.234924	0.759537

Critical Breach Schedule-Months by Team Experience Level Average



Contingency Table

RQ#3 Stable without a Critical Breach Schedule-Months

RQ#3 Team Experience Level Average IV High		0	1	Total
	Count			
	Expected			
High Experience		11	11	22
		6.65116	15.3488	
Not High		67	169	236
		71.3488	164.651	
Total		78	180	258

Tests

N	DF	-LogLike	RSquare (U)
258	1	2.0617956	0.0130

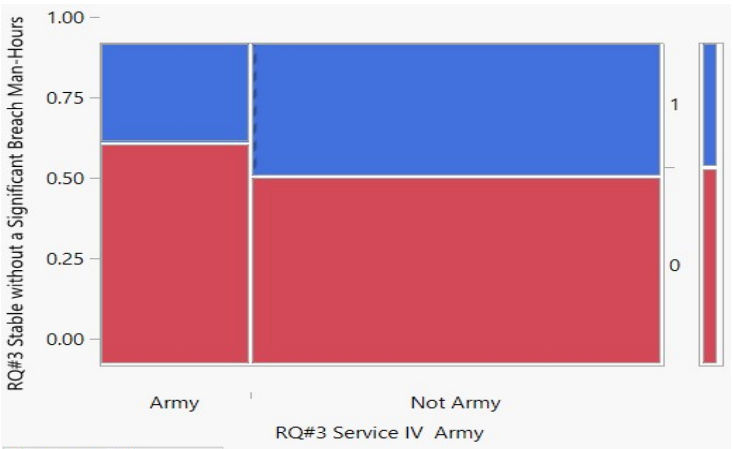
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	4.124	0.0423*
Pearson	4.456	0.0348*

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.9886	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is greater for RQ#3 Team Experience Level Average IV High=High Experience than Not High
Right	0.0342*	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is greater for RQ#3 Team Experience Level Average IV High=Not High than High Experience
2-Tail	0.0501	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is different across RQ#3 Team Experience Level Average IV High

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
2.522388	1.043805	6.095433

Critical Breach Schedule-Months by Team Experience Level High



Contingency Table

RQ#3 Stable without a Significant Breach Man-Hours

RQ#3 Service IV Army	Count / Expected	0	1	Total
Army		62	28	90
		55.2395	34.7605	
Not Army		143	101	244
		149.76	94.2395	
Total		205	129	334

Tests

N	DF	-LogLike	RSquare (U)
334	1	1.4943410	0.0067

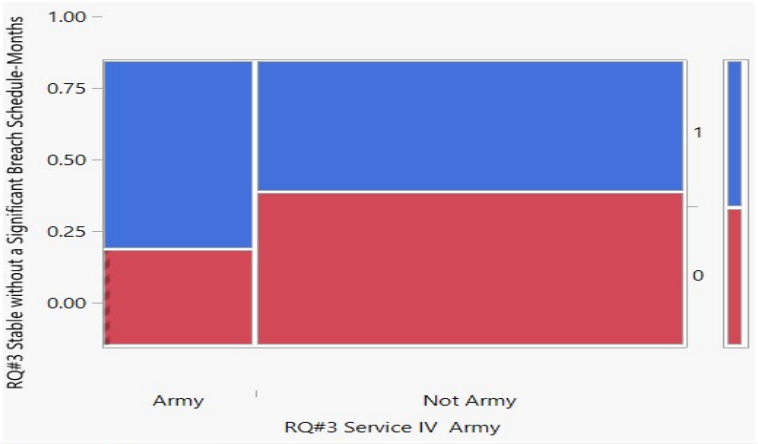
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	2.989	0.0838
Pearson	2.932	0.0868

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.9680	Prob(RQ#3 Stable without a Significant Breach Man-Hours=1) is greater for RQ#3 Service IV Army=Army than Not Army
Right	0.0555	Prob(RQ#3 Stable without a Significant Breach Man-Hours=1) is greater for RQ#3 Service IV Army=Not Army than Army
2-Tail	0.1000	Prob(RQ#3 Stable without a Significant Breach Man-Hours=1) is different across RQ#3 Service IV Army

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
1.563936	0.935519	2.61448

Significant Breach Man-Hours by Service Army



Contingency Table

RQ#3 Stable without a Significant Breach Schedule-Months

RQ#3 Service IV Army	Count / Expected	0	1	Total
Army	Count	27	53	80
	Expected	38.9474	41.0526	
Not Army	Count	121	103	224
	Expected	109.053	114.947	
Total		148	156	304

Tests

N	DF	-LogLike	RSquare (U)
304	1	4.9214298	0.0234

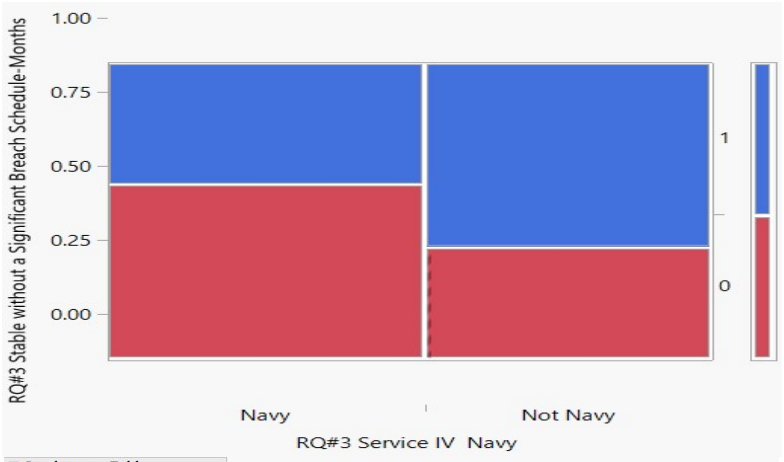
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	9.843	0.0017*
Pearson	9.693	0.0019*

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.0013*	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is greater for RQ#3 Service IV Army=Army than Not Army
Right	0.9995	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is greater for RQ#3 Service IV Army=Not Army than Army
2-Tail	0.0026*	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is different across RQ#3 Service IV Army

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
0.43365	0.254553	0.738756

Significant Breach Schedule-Months by Service Army



Contingency Table

RQ#3 Stable without a Significant Breach Schedule-Months

RQ#3 Service IV Navy	Count / Expected	0	1	Total
	Navy	94	66	160
		77.8947	82.1053	
	Not Navy	54	90	144
		70.1053	73.8947	
	Total	148	156	304

Tests

N	DF	-LogLike	RSquare (U)
304	1	6.9054734	0.0328

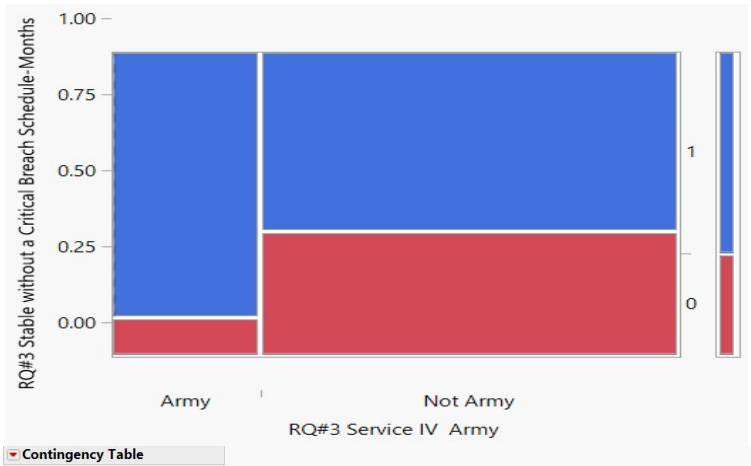
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	13.811	0.0002*
Pearson	13.699	0.0002*

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.9999	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is greater for RQ#3 Service IV Navy=Navy than Not Navy
Right	0.0002*	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is greater for RQ#3 Service IV Navy=Not Navy than Navy
2-Tail	0.0002*	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is different across RQ#3 Service IV Navy

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
2.373737	1.496403	3.76545

Significant Breach Schedule-Months by Service Navy



Contingency Table

RQ#3 Stable without a Critical Breach Schedule-Months

RQ#3 Service IV Army	Count / Expected	0	1	Total
Army		10	70	80
		26.8421	53.1579	
Not Army		92	132	224
		75.1579	148.842	
Total		102	202	304

Tests

N	DF	-LogLike	RSquare (U)
304	1	12.143194	0.0626

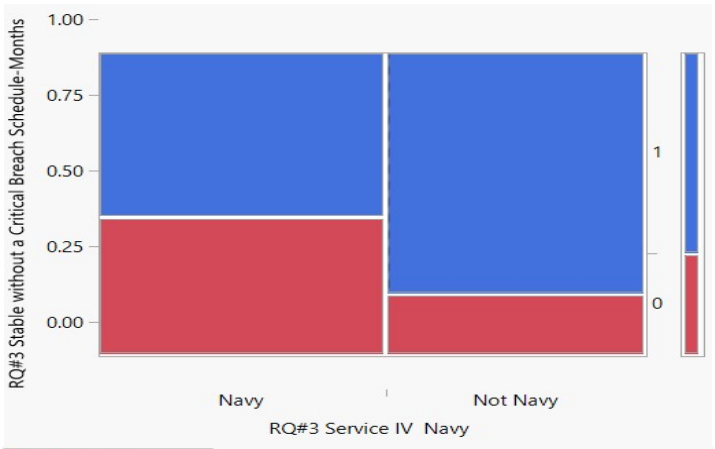
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	24.286	<.0001*
Pearson	21.584	<.0001*

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	<.0001*	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is greater for RQ#3 Service IV Army=Army than Not Army
Right	1.0000	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is greater for RQ#3 Service IV Army=Not Army than Army
2-Tail	<.0001*	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is different across RQ#3 Service IV Army

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
0.204969	0.100364	0.418601

Critical Breach Schedule-Months by Service Army



Contingency Table

RQ#3 Service IV Navy	Count / Expected	RQ#3 Stable without a Critical Breach Schedule-Months: 0	1	Total
Navy	Count	73	87	160
	Expected	53.6842	106.316	
Not Navy	Count	29	115	144
	Expected	48.3158	95.6842	
Total		102	202	304

Tests

N	DF	-LogLike	RSquare (U)
304	1	11.334513	0.0584

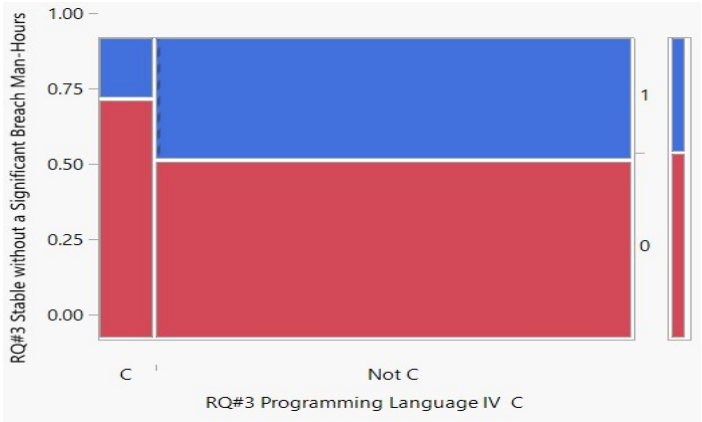
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	22.669	<.0001*
Pearson	22.081	<.0001*

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	1.0000	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is greater for RQ#3 Service IV Navy=Navy than Not Navy
Right	<.0001*	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is greater for RQ#3 Service IV Navy=Not Navy than Navy
2-Tail	<.0001*	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is different across RQ#3 Service IV Navy

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
3.327388	1.993114	5.554882

Critical Breach Schedule-Months by Service Navy



Contingency Table

RQ#3 Stable without a Significant Breach Man-Hours

RQ#3 Programming Language IV C	Count / Expected	0	1	Total
C		28	7	35
		21.5888	13.4112	
Not C		170	116	286
		176.411	109.589	
Total		198	123	321

Tests

N	DF	-LogLike	RSquare (U)
321	1	3.0315065	0.0142

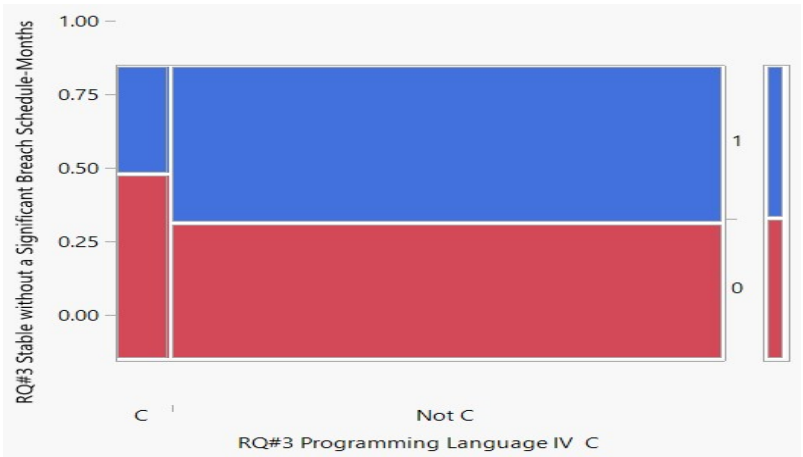
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	6.063	0.0138*
Pearson	5.577	0.0182*

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.9960	Prob(RQ#3 Stable without a Significant Breach Man-Hours=1) is greater for RQ#3 Programming Language IV C=C than Not C
Right	0.0125*	Prob(RQ#3 Stable without a Significant Breach Man-Hours=1) is greater for RQ#3 Programming Language IV C=Not C than C
2-Tail	0.0258*	Prob(RQ#3 Stable without a Significant Breach Man-Hours=1) is different across RQ#3 Programming Language IV C

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
2.729412	1.153584	6.457866

Significant Breach Man-Hours by Programming Language C



Contingency Table

RQ#3 Stable without a Significant Breach Schedule-Months

RQ#3 Programming Language IV C	Count / Expected	0	1	Total
C		17	10	27
		12.9897	14.0103	
Not C		123	141	264
		127.01	136.99	
Total		140	151	291

Tests

N	DF	-LogLike	RSquare (U)
291	1	1.3240156	0.0066

Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	2.648	0.1037
Pearson	2.630	0.1049

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.9664	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is greater for RQ#3 Programming Language IV C=C than Not C
Right	0.0777	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is greater for RQ#3 Programming Language IV C=Not C than C
2-Tail	0.1112	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is different across RQ#3 Programming Language IV C

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
1.94878	0.860304	4.414423

Significant Breach Schedule-Months by Programming Language C

Contingency Table

RQ#3 Programming Language IV Ada	RQ#3 Stable without a Significant Breach Schedule-Months	0	1	Total
Ada	Count	17	33	50
	Expected	24.055	25.945	
Not Ada	Count	123	118	241
	Expected	115.945	125.055	
Total		140	151	291

Tests

N	DF	-LogLike	RSquare (U)
291	1	2.4495028	0.0122

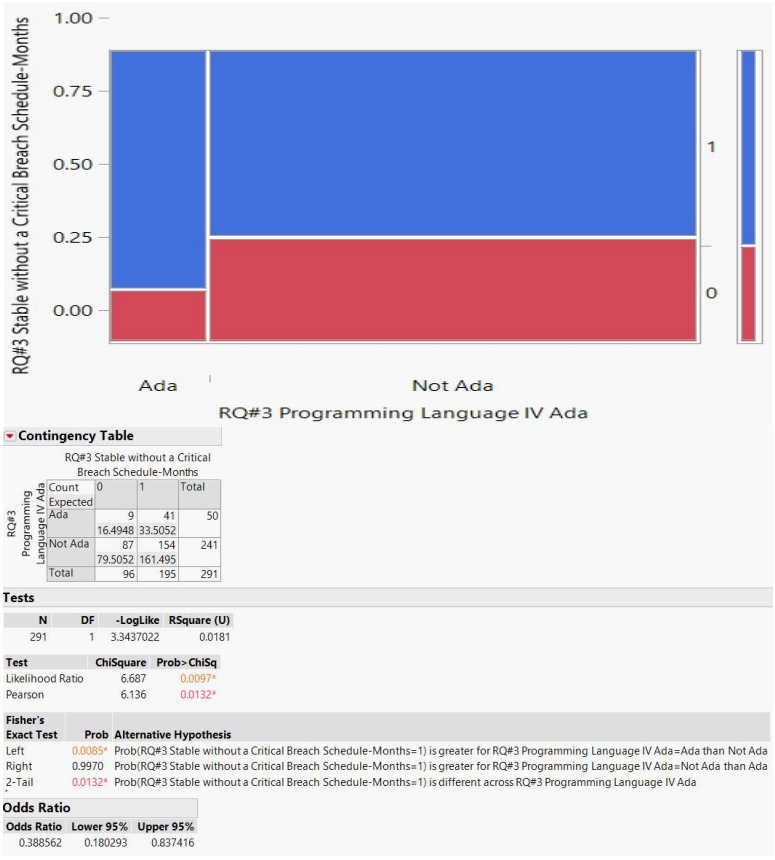
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	4.899	0.0269*
Pearson	4.815	0.0282*

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.0202*	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is greater for RQ#3 Programming Language IV Ada=Ada than Not Ada
Right	0.9911	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is greater for RQ#3 Programming Language IV Ada=Not Ada than Ada
2-Tail	0.0303*	Prob(RQ#3 Stable without a Significant Breach Schedule-Months=1) is different across RQ#3 Programming Language IV Ada

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
0.49421	0.261296	0.934741

Significant Breach Schedule-Months by Programming Language Ada



Contingency Table

RQ#3 Stable without a Critical Breach Schedule-Months

RQ#3 Programming Language IV Ada	Count / Expected	0	1	Total
Ada		9	41	50
		16.4948	33.5052	
Not Ada		87	154	241
		79.5052	161.495	
Total		96	195	291

Tests

N	DF	-LogLike	RSquare (U)
291	1	3.3437022	0.0181

Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	6.687	0.0097*
Pearson	6.136	0.0132*

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.0085*	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is greater for RQ#3 Programming Language IV Ada=Ada than Not Ada
Right	0.9970	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is greater for RQ#3 Programming Language IV Ada=Not Ada than Ada
2-Tail	0.0132*	Prob(RQ#3 Stable without a Critical Breach Schedule-Months=1) is different across RQ#3 Programming Language IV Ada

Odds Ratio

Odds Ratio	Lower 95%	Upper 95%
0.388562	0.180293	0.837416

Critical Breach Schedule-Months by Programming Language Ada

Bibliography

Abba, Wayne (2008). *The Trouble with Earned Schedule*. The Measurable News, Fall, Issue 4: 28-30.

A. Alshamrani and A. Bahattab (2015). *A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model*. (in English), International Journal of Computer Science Issues (IJCSI), vol. 12, no. 1, pp. 106-111, Jan 2015 2015-03-05 2015

Akbar, M. A., Jun, S., Khan, A. *Improving the Quality of Software Development Process by Introducing a New Methodology–AZ-Model*. in IEEE Access, vol. 6, pp. 4811-4823, 2018, doi: 10.1109/ACCESS.2017.2787981.

Borysowich, C. *Observations from a Tech Architect: Enterprise Implementation Issues & Solutions – Effort Distribution Across the Software Lifecycle*. Enterprise Architecture and EAI Blog.

Christensen, David S. and Scott R. Heise (1993). *Cost Performance Index Stability*. National Contract Management Journal, 25:7-15.

Christensen, David S., Payne, Kirk (1992). *Cost Performance Stability – Fact or Fiction?.* Journal of Parametrics, 10:27-40.

Chowdhury, A. E., Bhowmik, A., Hasan, H., and Rahim, M. S. (2017). *Analysis of the veracities of industry used software development life cycle methodologies*. AJSE, vol. 16, no. 2, pp. 1-8.

Clayson, D. S., Thal, J. A (2018). *Cost performance index stability: Insights from environmental remediation projects*. Journal of Defense Analytics and Logistics, 2(2), 94-109. doi:10.1108/jdal-11-2017-0024

Coopers, Lybrand and TASC. (1994). *The DoD regulatory cost premium: A quantitative assessment.* Washington, DC

Defense Acquisition University (2013). *Defense Acquisition Guidebook*. Washington, DC.

Defense Science Board. (2018). *Design and Acquisition of Software for Defense Systems*. Washington D.C.

Department of Defense. (1996). *DoD 5000.2–R: Management procedures for major defense acquisition programs.* Washington, DC.

Department of Defense. (1997). *Cost driver No. 3—Cost/schedule control system criteria (C/SCSC). DoD updated compendium of office of primary responsibility reports.* Washington, DC: DoD Regulatory Cost Premium Working Group.

Defense Contract Management Agency (2018). *DoD Earned Value Management Implementation Guide (EVMIG)*. Virginia: DCMA.

Department of Defense (2007). *Cost and Software Data Reporting (CSDR) Manual*. Washington DC.

Department of Defense, Chief Information Officer. (2019). *DoD Enterprise DevSecOps Reference Design*. Washington DC.

DoD 5000.04–M–1. Washington: DoD.

Ehley, B. (2013). *Another Failed Gov't Tech Project Cost $1.1 Billion*.

Fleming, Quentin W. and Joel M. Koppelman (2000). *Earned Value Project Management (2nd Edition). Pennsylvania: Project Management Institute*.

Garousi, V., Tarhan, A., Pfahl, D., Coşkunçay, A., and Demirörs, O. (2018). *Correlation of critical success factors with success of software projects: an empirical investigation*. Software Quality Journal, 27(1), 429-493. doi: 10.1007/s11219-018-9419-5

General Accounting Office. (1997). *Significant changes in DoD's earned value management process* (NSIAD 97-108). Washington, DC.

Heise, Capt Scott R. (1991). *A Review of Cost Performance Index Stability*. MS thesis, AFIT/GCA/LSY/91S-12. School of Systems and Logistics, Air Force Institute of Technology, Wright-Patterson AFB, OH.

Henderson, K. and Zwikael, O. (2008). *"Does Project Performance Stability Exist? A Re-examination of CPI and Evaluation of SPI(t) Stability."* Cross Talk, the Journal of Defense Software Engineering, 21(4): 7–13.

Kanaracus, Chris (2012). *Air Force scraps massive ERP project after racking up $1B in costs*. Computerworld.

Kannan, V., Jhajharia, S., Verma, S. (2014). *Agile vs waterfall: A Comparative Analysis*, IJSETR, Volume 3, Issue 10.

Payne, Kirk I. (1990). *An Investigation of the Stability of the Cost Performance Index*. MS thesis, AFIT/GCA/LSY/90S-6. School of Systems and Logistics, Air Force Institute of Technology, Wright-Patterson AFB, OH.

Petter, J. (2014). *An Analysis of Stability Properties in Earned Value Managements Cost Performance Index and Earned Schedules Schedule Performance Index*. Air Force Institute of Technology.

Petter, J.L., Ritschel, J.D. and White, E.D. III (2015), *"Stability properties in department of defense contracts: answering the controversy"*, Journal of Public Procurement, Vol. 15 No. 3, pp. 341-364

Purna Sudhakar, G. (2012). *A model of critical success factors for software projects*. Journal of Enterprise Information Management, 25(6), 537-558. doi: 10.1108/17410391211272829

Matkovic, P. and Tumbas, P. (2010). *A Comparative Overview of the Evolution of Software Development Models*. Journal of Industrial Engineering and Management, 1(4), 163-172.

McQuade, J. M. (2019). *Software Is Never Done: Refactoring the Acquisition Code for Competitive Advantage*. Defense Innovation Board. available at https://innovation.defense.gov.

Lanham, N., Russo, M., Strickland, D., Cipressi, R., Palmer, S., & Rudloff, C. (2018). *Department of Defense Software Resource Data Report (SRDR) Verification and Validation (V&V) Guide Version 4.0*.

Lavazza, L., Morasca, S., & Tosi, D. (2018). An Empirical Study on the Factors Affecting Software Development Productivity. *e-Informatica Software Engineering Journal*.

Lipke, Walt. (2013). *Schedule is Different*. The Measurable News, March & Summer 2003. http://www.earnedschedule.com/Docs/Schedule%20is%20Different.pdf

Nasir, M. H. N. and Sahibuddin, S. (2011). *Critical success factors for software projects: A comparative study*. Scientific Research and Essays, 6(10), 2174-2186. doi:10.5897/SRE10.1171

Osd Cape. (2019). *The Software Resources Data Report (SRDR) Implementation Guidance*. Washington DC.

Schwartz, Moshe (2010). *The Nunn-McCurdy Act: Background, Analysis, and Issues for Congress.* Congressional Research Service

Tishler, A., Dvir, D., Shenhar, A., and Lipovetsky, S. (1996). *Identifying critical success factors in defense development projects: A multivariate analysis*. Technological Forecasting and Social Change, 51(2), 151-171. doi: 10.1016/0040-1625(95)00197-2

Vanderbyl, S. and Kobelak, S. (2007). *Critical success factors for biotechnology industry in Canada*. Journal of Commercial Biotechnology, 13(2), 68-77. doi: 10.1057/palgrave.jcb.3050042

Yang, Y., et al. (2008). *Phase Distribution of Software Development Effort*. Empirical Software Engineering and Measurement. October 2008.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
25-03-2021	Master's Thesis	May 2019 – March 2021

TITLE AND SUBTITLE

An Analysis of Stability in Software Resource Data Report (SRDR) Programs Computer Software Configuration Items (CSCI)

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Violette, Trevor J., 1st Lieutenant, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way, Building 640
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENV-MS-21-M-280

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AGENCY (spelled out) Norwegian Defence Research Establishment
ADDRESS Norwegian Defence Research Establishment (FFI) PB 25, 2027 Kieller
PHONE and EMAIL 63 80 77 82 Helene.Berg@ffi.no
ATTN: POC
Helene Berg

10. SPONSOR/MONITOR'S ACRONYM(S)

NATO

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRUBTION STATEMENT A. APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES
This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

This research studies cost and schedule stability in programs that utilize Software Resource Data Report (SRDR) reporting standards. We find software programs at the Computer Software Configuration Item (CSCI) level show much lower levels of stability than previously published DoD stability research that focused on aircraft. A comparison of software development methods found little to no difference between Agile and Plan Driven methodologies. Critical Success Factors (CSF) were identified from prior literature and used to examine CSCIs from the SRDR dataset. Focusing on schedule or cost resulted in different variables showing significance. A CSCI is more likely to remain on budget when using a team with a low level of average experience and being judicious in your contractor selection. A CSCI is more likely to finish on schedule when a team has an average level of experience and Boeing is used as the primary contractor. A CSCI is more likely to remain on budget and on schedule when Lockheed Martin is the lead contractor and the CSCI is programmed in any language other than C. This research can be used by program managers and cost analysts to identify the critical success factors that can be utilized in the Department of Defense software environment to create trade off space between cost and schedule.

15. SUBJECT TERMS
SRDR Programs, Cost Stability, Schedule Stability, Contingency Table Analysis

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Jonathan D. Ritschel, AFIT/ENV
U	U	U	UU	122	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-3636, ext 4484 (NOT DSN) (Jonathan.Ritschel@afit.edu)

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18